СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра інформаційних технологій та програмування

Пояснювальна записка

до магістерської дипломної роботи
магістр
(освітньо-кваліфікаційний рівень)
на тему Проектування інформаційної системи для автоматизованої торгівлі

Виконав: студент 2 курсу, групи ІСТ-23зм
126 «Інформаційні системи та технології»
(шифр і назва спеціальності)
Комягін Д. В.
(прізвище та ініціали)
Керівник Лифар В. О.
(прізвище та ініціали)
Рецензент Меняйленко О.С.
(прізвище та ініціали)

Київ – 2024 року

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ ДО МАГІСТЕРСЬКОЇ ДИПЛОМНОЇ РОБОТИ

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

Освітньо-кваліфікаційний рівень магістр

Спеціальність 126 «Інформаційні системи та технології»

(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТП
_____ д.т.н., проф., Захожай О. І.
(підпис)
«____»_____2024р.

ЗАВДАННЯ
на магістерську дипломну роботу студенту
Комягіну Дмитру Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Проектування інформаційної системи для автоматизованої торгівлі
керівник роботи Лифар Володимир Олексійович, д.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу
від «06» 12 2024року №361/15.15-С
2. Строк подання студентом роботи 15.12.2024
3. Вихідні дані до роботи: Матеріали науково-дослідної практики, науково-методична література; дані інтернет-мережі .
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   4.1 Вступ
   4.2 Аналітична частина
   4.3 Обґрунтування проектних рішень та моделі системи.
   4.4 Створення автоматизованої торгової системи.
   4.5 Висновки
   4.6 Перелік використаних джерел

5. Перелік графічного матеріалу (з точним значенням обов'язків креслень)_____

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 18 жовтня 2024

КАЛЕНДАРНИЙ ПЛАН

№ з\п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної галузі	25.10.2 – 28.10.24	
2	Пошук та аналіз існуючих рішень	29.10.24 – 04.11.24	
3	Аналіз алгоритмів	05.11.24 – 10.11.24	
4	Розробка алгоритму	11.11.24 – 22.11.24	
5	Інтеграція алгоритму до біржі	24.11.24 – 24.11.24	
6	Тестування алгоритму	24.11.24 – 02.12.24	
7	Оформлення пояснювальної записки	02.12.24 – 05.12.24	
8	Підготовка та подання магістерської роботи до захисту	06.12.24 – 06.12.24	

Студент_____ Комягін Д.В.
(підпис) (прізвище та ініціали)

Керівник роботи_____ Лифар В.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська дипломна робота: 64 стор., 15 рис., 4 таб., 10 джерел.

Мета роботи – Метою даної дипломної роботи є розробка інформаційної системи для автоматизації торгових операцій, що підвищить ефективність процесів та мінімізує ризики в умовах сучасних ринків. Основна увага приділяється створенню інструменту, який спростить роботу трейдерів шляхом автоматизації аналізу ринкових даних, виконання торгових рішень та моніторингу ризиків.

Об’єкт дослідження Об'єктом дослідження є процеси автоматизованої торгівлі, що включають використання програмних систем для аналізу даних, виконання угод та управління фінансовими ризиками.

Завдання дослідження:

1. Проаналізувати сучасні методи автоматизації торгівлі, їх переваги, недоліки та обмеження.

2. Розробити архітектуру інформаційної системи, що забезпечить автоматизацію ключових процесів торгівлі.

3. Реалізувати програмне забезпечення для автоматизованого виконання торгових операцій та моніторингу ризиків.

4. Провести експериментальне тестування розробленої системи на основі симуляції реальних ринкових сценаріїв.

5. Здійснити порівняльний аналіз результатів експериментального тестування з існуючими аналогами.

Наукова новизна:

Наукова новизна дослідження полягає у розробці інформаційної системи для автоматизованої торгівлі, що поєднує високий рівень автоматизації з інтеграцією механізмів управління ризиками. Система пропонує адаптивний підхід до вибору торгових стратегій, використовуючи сучасні методи аналізу

даних, такі як машинне навчання та алгоритми оптимізації.

Додатковою перевагою є модульна архітектура системи, що дозволяє масштабування та адаптацію до різних умов ринку. Розроблене рішення має потенціал стати основою для впровадження інтелектуальних систем підтримки прийняття рішень у фінансовій сфері.

АВТОМАТИЗАЦІЯ, АЛГОРИТМІЧНА ТОРГІВЛЯ, ЕФЕКТИВНІСТЬ, ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, РИЗИК-МЕНЕДЖМЕНТ, ІНФОРМАЦІЙНА СИСТЕМА, СИМУЛЯЦІЯ, ТРЕЙДИНГ, УПРАВЛІННЯ РИЗИКАМИ.

ЗМІСТ

Вступ 5
1 Аналітична частина 6
1.1 Аналіз ринку автоматизованої торгівлі: сучасні тенденції та перспективи 6
1.2 Аналіз методів і технологій автоматизованої торгівлі 8
1.3 Обґрунтування вибору торгової стратегії Moving Average 10
1.4 Постановка задачі 13
2 Обґрунтування проектних рішень та моделі системи 15
2.1 Обґрунтування проектних рішень з інформаційного 15
2.2 Обґрунтування проектних рішень щодо функціонального 16
2.3 Обґрунтування вибору програмних засобів 17
2.4 Функціональна модель системи 19
2.5 Інформаційна модель системи 23
2.6 Визначення функцій автоматизованої торгової системи 24
2.7 Оптимізація процесів автоматизованої торгівлі 25
3 Створення автоматизованої торгової системи 28
3.1 Підготовка до роботи автоматизованої торгової системи 29
3.2 Тестування торгового радника 31
3.3 Оптимізація параметрів торгового радника 33
3.4 Перевірка працездатності розробленої торгової системи 34
3.5 Розгортання торгової системи на реальному рахунку 36
Висновок 39
Список використаних джерел 41
Додаток 42

Вступ

В умовах динамічного розвитку цифрових технологій та зростання обсягів даних фінансових ринків автоматизація торгівельних процесів стає не лише бажаною, а й необхідною. Зміна поведінки ринку, висока конкуренція та потреба у швидкому прийнятті рішень вимагають застосування сучасних інформаційних технологій, здатних забезпечити точність, оперативність і адаптивність торгових операцій.

Актуальність даного дослідження обумовлена зростанням ролі програмних систем у торгівлі, що дозволяють мінімізувати людський фактор, підвищити ефективність процесів і зменшити ризики.

Мета магістерської дипломної роботи полягає у розробці інформаційної системи, яка забезпечить автоматизацію ключових етапів торгових операцій, включаючи аналіз ринкових даних, виконання торгових рішень та управління ризиками.

У рамках роботи здійснено теоретичний аналіз основних підходів до автоматизації торгівлі, розглянуто сучасні інструменти та технології, що застосовуються для цього. На основі отриманих результатів запропоновано програмне рішення, яке відповідає потребам ринку та відображає інноваційні підходи до інтеграції інформаційних технологій у торгівлю.

Особливість роботи полягає у створенні модульної архітектури програмної системи, яка дозволяє адаптувати її під конкретні торгові стратегії. На завершальному етапі проведено експериментальне тестування системи з використанням симуляцій реальних ринкових умов, що забезпечило можливість оцінити її ефективність та практичну придатність.

Таким чином, дана робота робить внесок у розвиток сучасних інформаційних технологій у фінансовій сфері, сприяючи підвищенню надійності та ефективності автоматизованих торгових процесів.

1 Аналітична частина

1.1 Аналіз ринку автоматизованої торгівлі: сучасні тенденції та перспективи

Автоматизована торгівля, або алгоритмічна торгівля, є важливою складовою фінансових ринків у сучасному світі. Завдяки розвитку технологій, трейдери та інвестори все частіше використовують торгових радників, які здатні швидко аналізувати великі обсяги даних і приймати торгові рішення без участі людини. Цей підхід забезпечує високу швидкість реакції на ринкові зміни та знижує вплив людського фактора на результат торгівлі.

Розвиток автоматизованої торгівлі нерозривно пов’язаний із прогресом у сфері інформаційних технологій. Згідно з даними досліджень, проведених провідними аналітичними компаніями, понад 70% торгівлі на фондових ринках у США здійснюється за допомогою алгоритмів. Цей показник демонструє важливість автоматизації у фінансовій сфері та підтверджує її ключову роль у сучасному інвестуванні.

Автоматизовані системи торгівлі дозволяють не лише оптимізувати процеси, але й відкривають нові можливості для індивідуальних інвесторів. Наприклад, за допомогою таких платформ, як MetaTrader 4 або MetaTrader 5, користувачі можуть створювати власні алгоритми та тестувати їх на історичних даних.

Однією з ключових тенденцій у розвитку автоматизованої торгівлі є інтеграція технологій штучного інтелекту (ШІ). Сучасні алгоритми здатні використовувати машинне навчання для виявлення складних закономірностей на ринку, що робить їх більш ефективними порівняно з традиційними методами аналізу. Наприклад, штучні нейронні мережі можуть враховувати такі фактори, як настрої інвесторів, новини та навіть мікроструктуру ринку, забезпечуючи при цьому високу точність прогнозів.

Ще однією важливою тенденцією є поширення високочастотної торгівлі (HFT). Цей підхід базується на використанні потужних серверів і спеціалізованого програмного забезпечення для виконання мільйонів операцій у найкоротші проміжки часу. Однак HFT також викликає чимало дискусій через його потенційний вплив на стабільність ринків.

Також варто відзначити зростаючу роль хмарних технологій у автоматизованій торгівлі. Використання хмарних платформ дозволяє трейдерам знижувати витрати на технічну інфраструктуру, одночасно забезпечуючи доступ до сучасних інструментів аналізу та тестування.

Серед основних переваг автоматизованої торгівлі можна виділити наступні:

1. Швидкість виконання операцій. Алгоритми здатні миттєво реагувати на ринкові зміни, що особливо важливо у періоди високої волатильності.

2. Об’єктивність рішень. Алгоритми усувають людські емоції з процесу прийняття рішень, що знижує ризик помилок, пов’язаних із психологічними факторами.

3. Можливість тестування. Автоматизовані системи дозволяють проводити бектестинг стратегій на історичних даних, що дає змогу оцінити їх ефективність перед використанням у реальних умовах.

Однак автоматизована торгівля має й певні недоліки. Наприклад, системи можуть бути вразливими до технічних збоїв або неочікуваних ринкових подій. Крім того, використання складних алгоритмів часто потребує значних знань у програмуванні та фінансовій аналітиці, що може бути недоступним для багатьох інвесторів.

Перспективи автоматизованої торгівлі пов’язані зі зростаючою інтеграцією передових технологій. Наприклад, очікується, що роль ШІ у цій сфері буде продовжувати зростати, забезпечуючи більш точні прогнози та адаптивні алгоритми. Крім того, розвиток технологій блокчейн може створити нові можливості для автоматизації угод на децентралізованих платформах.

Зростання обсягу даних, доступних для аналізу, також сприятиме вдосконаленню автоматизованих систем. Великі дані (Big Data) дозволяють враховувати безліч факторів, які раніше залишалися поза увагою трейдерів, забезпечуючи більш глибоке розуміння ринкових тенденцій.

Аналіз сучасного стану та перспектив автоматизованої торгівлі підтверджує її важливу роль у фінансовій сфері. Інтеграція нових технологій, таких як штучний інтелект і блокчейн, створює значні можливості для подальшого розвитку цієї галузі. Однак трейдерам та інвесторам необхідно враховувати ризики, пов’язані з використанням автоматизованих систем, і постійно адаптувати свої стратегії до змінних ринкових умов..

1.2 Аналіз методів і технологій автоматизованої торгівлі

Автоматизовані торгові системи (АТС) є важливим елементом сучасних фінансових ринків, що дозволяють здійснювати операції без постійного втручання трейдера. Ці системи застосовують різноманітні алгоритми та стратегії для аналізу ринкових даних і прийняття рішень щодо торгівлі активами. Ефективність їх роботи базується на математичних методах, потужності обчислювальної техніки та гнучкості налаштувань під специфіку ринку [1].

Основна перевага АТС полягає в усуненні емоційного впливу на процес прийняття рішень. Людські емоції часто стають причиною помилок у торгівлі, зокрема через надмірний ризик або передчасний вихід із позиції. Алгоритми, у свою чергу, діють суворо за заданими параметрами, забезпечуючи сталість і об'єктивність у процесі аналізу ринку [2].

Технічний аналіз є ключовим інструментом у розробці алгоритмів для АТС. Його основна ідея полягає в тому, що ринкові ціни відображають усю необхідну інформацію, а історичні дані можуть бути використані для прогнозування майбутніх змін. До основних методів технічного аналізу належать:

1. Ковзні середні (Moving Averages, MA): цей метод визначає середню ціну активу за певний період і застосовується для згладжування коливань ціни. Використання ковзних середніх у АТС дозволяє фільтрувати ринковий шум і визначати тренди [4].

2. Стохастичний осцилятор: інструмент для оцінки перекупленості або перепроданості активу. У поєднанні з іншими індикаторами він є важливим елементом багатьох торгових стратегій [3].

3. Індекс відносної сили (Relative Strength Index, RSI): метрика, що оцінює силу поточного тренду і його ймовірний розворот. RSI інтегрується у алгоритми для ідентифікації зон перевищення купівельного або продажного тиску [5].

З розвитком комп'ютерних технологій традиційні алгоритми автоматизованої торгівлі поступово доповнюються методами штучного інтелекту (ШІ) та машинного навчання (МН). ШІ дозволяє адаптуватися до мінливих умов ринку, аналізувати великі масиви даних у режимі реального часу та прогнозувати складні патерни [6].

Методи машинного навчання, такі як нейронні мережі, використовуються для прогнозування ринкових цін. Наприклад, рекурентні нейронні мережі (RNN) ефективно аналізують часові ряди, що є характерним для фінансових даних. Інші популярні методи включають підтримку векторних машин (SVM) і алгоритми кластеризації, які можуть визначати закономірності у ринкових даних [8].

Попри значні переваги, автоматизовані торгові системи мають низку обмежень:

1. Перенавчання моделей: складні алгоритми машинного навчання інколи стають занадто адаптованими до історичних даних, що може призвести до зниження їхньої ефективності в умовах реального ринку [10].

2. Залежність від технічної інфраструктури: для ефективної роботи АТС потрібні високопродуктивні сервери, надійне інтернет-з'єднання та

сучасне програмне забезпечення. Будь-які технічні збої можуть негативно вплинути на результати торгівлі [7].

3. Проблема адаптації: алгоритми можуть бути недостатньо гнучкими для швидкого реагування на раптові зміни ринкових умов, такі як економічні кризи або політичні події [9].

Аналіз показує, що автоматизовані торгові системи є важливим інструментом для роботи на фінансових ринках. Їхнє використання дозволяє підвищити ефективність торгівлі завдяки швидкості, точності та усуненню емоційного впливу. Однак для створення конкурентоспроможної АТС необхідно враховувати сучасні методи аналізу, впроваджувати інноваційні технології та долати існуючі обмеження

1.3 Обґрунтування вибору торгової стратегії Moving Average

Торгова стратегія на основі ковзних середніх (Moving Average, MA) є однією з найпопулярніших у фінансовій торгівлі завдяки простоті реалізації та ефективності у визначенні ринкових трендів. Основний принцип стратегії полягає у використанні середнього значення цін за певний період для згладжування короткострокових коливань та виявлення основного напряму руху ринку [1].

Ковзні середні бувають кількох типів, зокрема:

1. Проста ковзна середня (Simple Moving Average, SMA): розраховується як середнє арифметичне цін за визначений період. Наприклад, 10-денна SMA представляє середню ціну активу за останні 10 днів.

2. Експоненціальна ковзна середня (Exponential Moving Average, EMA): надає більшу вагу останнім значенням цін, що робить її чутливішою до нових змін на ринку.

3. Зважена ковзна середня (Weighted Moving Average, WMA): враховує різні вагові коефіцієнти для кожного дня в обраному періоді, що дозволяє

точніше враховувати поточні тенденції [9].

Стратегія Moving Average зазвичай використовується для виявлення двох основних торгових сигналів:

- Перетин ціни та ковзної середньої: якщо ціна перетинає МА знизу вгору, це сигнал до купівлі, а якщо зверху вниз — сигнал до продажу.

- Перетин швидких і повільних ковзних середніх: використання двох МА із різними періодами дозволяє отримувати точніші сигнали. Наприклад, якщо коротша EMA перетинає довшу знизу вгору, це свідчить про висхідний тренд.

Стратегія МА має кілька ключових переваг, які роблять її оптимальним вибором для автоматизованих торгових систем:

1. Простота реалізації: алгоритми, що базуються на МА, легко програмуються, а їхні результати легко інтерпретуються навіть для початківців. Завдяки цьому МА широко використовується у фінансових програмних системах [6].

2. Універсальність: Moving Average можна застосовувати до різних фінансових активів, таких як акції, валюти, сировинні товари тощо. Це робить її адаптованою до різних ринкових умов [3].

3. Можливість оптимізації: трейдер може легко змінювати параметри МА (період та тип) відповідно до свого стилю торгівлі або специфіки ринку, що підвищує ефективність алгоритму.

4. Фільтрація шуму: МА допомагає зменшити вплив короткострокових ринкових коливань, зосереджуючись на загальних тенденціях. Це особливо важливо для автоматизованих систем, які часто працюють із великим обсягом даних [2].

Попри численні переваги, стратегія МА має й деякі обмеження:

1. Затримка сигналів: через використання середнього значення ціни МА реагує на зміни із певною затримкою. Це може призвести до втрати частини прибутку або збільшення збитків під час швидких ринкових змін [6].

2. Неефективність у бічному тренді: МА погано працює у фазі

консолідації ринку, коли ціна коливається в межах вузького діапазону. Це може спричинити велику кількість помилкових сигналів.

3. Чутливість до параметрів: неправильний вибір періоду або типу MA може знизити її ефективність. Вирішенням цієї проблеми є використання оптимізаційних алгоритмів, таких як генетичні або методи машинного навчання, для підбору найкращих параметрів [4].

Серед безлічі торгових стратегій MA обрана як основа для автоматизованої торгової системи з таких причин:

1. Збалансованість між складністю та ефективністю: MA забезпечує прийнятний рівень ефективності за невеликої обчислювальної складності, що знижує вимоги до апаратного забезпечення [8].

2. Широка підтримка в програмному забезпеченні: Moving Average є стандартним інструментом у більшості торгових платформ і бібліотек, таких як MetaTrader, Python, і R, що полегшує інтеграцію [7].

3. Гнучкість у налаштуванні: MA дозволяє налаштовувати стратегію під різні ринкові умови завдяки можливості вибору типу ковзної середньої та її параметрів [10].

4. Доказова база: Багато досліджень підтверджують ефективність MA у визначенні трендів, що забезпечує її надійність як основи для розробки автоматизованої торгової системи [4].

Для обґрунтування вибору MA слід порівняти її з іншими поширеними стратегіями:

1. Стохастичний осцилятор: попри свою ефективність у зоні перепроданості/перекупленості, цей метод менш надійний у трендових ринках. MA, натомість, краще виявляє довгострокові тенденції.

2. Індекс відносної сили (RSI): RSI більш ефективний для визначення короткострокових змін, тоді як MA працює на довших інтервалах, що підходить для нашої мети [9].

3. Стратегії на основі свічкових патернів: ці стратегії вимагають більше обчислювальних ресурсів і часто піддаються впливу ринкового шуму, тоді як

MA забезпечує стабільність сигналів [9].

Таким чином, Moving Average є оптимальним вибором для розробки автоматизованої торгової системи. Ця стратегія забезпечує баланс між простотою, ефективністю та універсальністю. Вона є надійним інструментом для аналізу ринку та може бути адаптована до різних ринкових умов завдяки гнучким налаштуванням.

1.4 Постановка задачі

Торговий радник — це програмне забезпечення або повноцінна система, що забезпечує автоматизацію торгових операцій і підвищення прибутковості роботи на фінансових ринках, таких як Forex. Метою створення торгового радника є полегшення процесу прийняття рішень трейдером і мінімізація впливу людського фактора на результати торгівлі [4].

Ключовим параметром ефективності радника є його здатність забезпечувати стабільний прибуток. Без цього інші переваги, такі як зручність використання, зниження емоційного напруження або можливість автоматизації рутинних процесів, втрачають свою значущість. Окрім прибутковості, торговий радник має відповідати низці додаткових вимог, що підвищують його надійність та продуктивність. Зокрема, радник повинен відповідати таким критеріям [3]:

- Швидкість реакції на зміну ціни: система має миттєво реагувати на коливання ринкових котирувань для уникнення втрати можливостей.

- Стабільність роботи: виключення збоїв, таких як зависання або циклічні помилки, що можуть призвести до зупинки торгівлі.

- Однозначність виконання алгоритмів: усі параметри системи та їхня обробка повинні бути чітко визначеними й уніфікованими, щоб уникнути неоднозначних рішень.

- Врахування ринкової психології: система має враховувати поведінку інших учасників ринку, зокрема автоматизованих систем великих фінансових

установ.

- Інтеграція базових концепцій технічного аналізу: алгоритми радника повинні враховувати ключові інструменти, такі як числа Фібоначчі, рівні підтримки та опору.

Зважаючи на ці вимоги, в рамках даної роботи необхідно створити торгового радника, що функціонуватиме на основі стратегії ковзної середньої (Moving Average). Особливу увагу приділено інтеграції механізмів адаптації до ринкових умов, регулюванню ризиків і можливості масштабування алгоритму для роботи з різними фінансовими інструментами.

Для досягнення мети поставлено такі завдання:

1. Проектування структури торгового радника: визначити основні компоненти системи, описати їх функції та взаємодію.

2. Розробка математичного апарату: створити або адаптувати моделі прийняття рішень для кожного компонента радника, зокрема алгоритми аналізу ринку та управління капіталом.

3. Реалізація алгоритмів: програмно реалізувати функціонал радника мовою MQL4, яка інтегрована в популярні торгові платформи, такі як MetaTrader 4.

4. Емпіричне дослідження: протестувати роботу радника на історичних даних та порівняти результати з ефективністю існуючих торгових систем.

5. Оптимізація параметрів: забезпечити можливість налаштування таких показників, як рівень ризику, торговий лот, періоди ковзної середньої тощо, для підвищення адаптивності системи.

Результатом реалізації проєкту стане повноцінний автоматизований радник, здатний приймати торгові рішення на основі аналізу ринкових даних, виконувати операції з високою точністю та забезпечувати прибутковість у різних ринкових умовах..

2 Обґрунтування проектних рішень та моделі системи

2.1 Обґрунтування проектних рішень з інформаційного забезпечення

Інформаційне забезпечення – це сукупність єдиної системи класифікації та кодування інформації, уніфікованих систем документації, схем інформаційних потоків, що циркулюють для підприємства, методологія побудови баз даних [6].

До інформаційного забезпечення пред'являються такі загальні вимоги [7]:

1) інформаційне забезпечення має бути достатнім для підтримки всіх функцій об'єкта, що автоматизуються;

2) для кодування інформації повинні використовуватися прийняті у замовника класифікатори;

3) для кодування вхідної та вихідної інформації, яка використовується на найвищому рівні управління, повинні бути використані класифікатори цього рівня;

4) повинна бути забезпечена сумісність з інформаційним забезпеченням систем, що взаємодіють із системою, що розробляється;

5) форми документів мають відповідати вимогам корпоративних стандартів замовника (або уніфікованої системи документації);

6) структура документів та екранних форм має відповідати характеристикам терміналів на робочих місцях кінцевих користувачів;

7) графіки формування та зміст інформаційних повідомлень, а також використовувані абревіатури повинні бути загальноприйняті у цій предметній галузі та узгоджені із замовником;

8) в ІВ мають бути передбачені засоби контролю вхідної та результатної інформації, оновлення даних в інформаційних масивах,

контролю цілісності інформаційної бази, захисту від несанкціонованого доступу.

У цьому роботі використовувалося інформаційне забезпечення, подане у таблиці 2.1.

Таблиця 2.1 - Інформаційне забезпечення

№	Найменування	Характеристика
1.	Мова програмування для Meta Trader	MQL4 (MetaQuotes Language) – це мова програмування, яка використовується для автоматизації своїх дій на валютному ринку та реалізації торгових стратегій у рамках торгового терміналу MetaTrader4.
2.	Forex EA Generator	Програма для створення радників та конструктор для індикаторів Meta Trader 4
3.	Tradecision Pro	Універсальна програма технічного аналізу, що не має аналогів за доступним набором функцій. відомих програм технічного аналізу
4.	Forex Strategy Builder	Програма для створення і тестування стратегій торгівлі на онлайн біржі Forex

2.2 Обґрунтування проектних рішень щодо функціонального забезпечення

Функціональні вимоги – це перелік сервісів, які має виконувати система, причому має бути зазначено, як система реагує ті чи інші вхідні дані, як поводиться у певних ситуаціях тощо. У деяких випадках зазначається, що система не повинна робити.

Функціональні вимоги визначають поведінку програмної системи, яка має бути створена розробниками для надання можливості виконання користувачами своїх обов'язків у рамках бізнес-вимог і в контексті вимог користувача [6].

Вимоги до функціонального забезпечення, що пред'являється до системи, що розробляється, представлені в таблиці 2.2.

Таблиця 2.2 - Функціональне забезпечення

№	Найменування	Характеристика
1.	Торгівля	Відкриття угод Закриття угод
2.	Розробка додатків	Розробка торгових радників та роботів
3.	Аналіз ринку	Фундаментальний аналіз Комп'ютерний аналіз Технічний аналіз Свічковий аналіз
4.	Пошук інвесторів	Пошук потенційних клієнтів та інвесторів

2.3 Обґрунтування вибору програмних засобів

Торгові радники Forex дають можливість автоматизувати процеси торгівлі та аналізу. Радники, які розроблені за допомогою мови програмування Meta Quotes Language 4 можуть дуже швидко здійснювати технічний аналіз актуальних цін. Також, враховуючи сигнали Форекс та прогнози, за допомогою радників можна побудувати власну

тактику під час управління своєї діяльності міжнародному валютному ринку [8].

Торгові радники можна створити різними способами відповідно до конкретної торгової стратегії. Існує безліч програмних засобів, які допомагають у створенні радників і роботів.

У таблиці 2.3 наведено порівняльну характеристику найбільш популярних:

– Forex EA Generator – програма для створення радників та конструктор для індикаторів Metatrader 4.

– Safir XP – нейропрограма з унікальною функціональністю. На відміну від інших нейропрограм Safir XP сам створює готову механічну торговельну систему з використанням нейромереж та нечіткої логіки – користувачеві потрібно лише вказати список з кількох індикаторів, на основі яких буде створюватись система – або використовуватись для покращення та оптимізації будь-якої вже наявної механічної торгової системи.

– MetaEditor є зручним, багатофункціональним середовищем розробки, інтегрованим з торговим терміналом. Він призначений для розробки програм на мові MetaQuotes Language 4 та 5 (MQL4 та MQL5). З його допомогою можна створювати радники (експерти), індикатори користувача та скрипти, які надалі будуть застосовуватися в клієнтському терміналі для аналітичних і торгових цілей.

– Tradecision Pro – універсальна програма технічного аналізу, яка не має аналогів доступного набору функцій. Включає практично весь функціонал найвідоміших програм технічного аналізу.

– Forex Strategy Builder – програма для створення та тестування стратегій торгівлі на онлайн біржі Forex. Forex Strategy Builder створює реальні умови для проведення тестової торгівлі, оперуючи справжніми ринковими даними.

Таблиця 2.3 - Порівняльна характеристика ПЗ

	Forex EA Generator	Safir FX	MetaEditor	Trader-cision Pro	Forex Strategy Builder
Не вимагає навичок програмування	+	-	-	-	-
Підтримує MetaTraider4	+	-	+	+	-
Є інтегрованою з MetaTraider4	-	-	+	-	-
Створює експертів	+	+	+	-	-
Встановлює властивості ордерів	+	-	+	-	-
Встановлює час виконання ордерів	+	-	+	-	-
Проводить тестування	-	-	+	-	+
Підтримує MQL 4	+	-	+	-	-

Отже, за даними таблиці отримуємо, що найзручнішим створення торгових радників є MetaEditor. Єдиним мінус цієї програми стало те, що для роботи в ній потрібні навички програмування, але завдяки цьому можна розширити функціонал радника і здійснювати угоди з більшою точністю.

2.4 Функціональна модель системи

Функціональна модель системи призначена для вивчення особливостей роботи системи та її призначення у взаємозв'язку із внутрішніми та зовнішніми елементами [2]. Функціональна модель системи, розроблена серед AllFusion Process Modeler, наведено рисунку 2.1.

Вхідними параметрами системи «ЯК ПОВИННО БУТИ» є: котирування валютної пари, стан рахунку та дані щодо відкритих угод; вихідними: звіт про стан рахунку, відкриття ордера, закриття ордера, оновлення ордера.

Радник, що розробляється, включає кілька компонентів, а саме: компонент аналізу та прогнозу, компонент управління, компонент побудови стратегії і компонент торгівлі, які в подальшому використовуються при роботі різних блоків радника. А керуючими стрілками є: правила торгівлі Форекс, бажана стратегія та метауправління на закриття позиції.

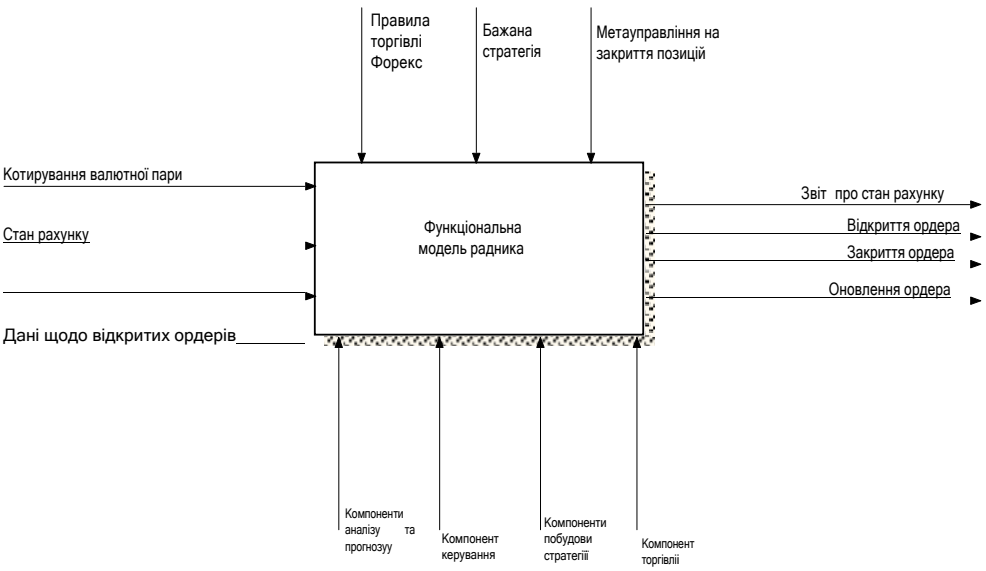

Рисунок 2.1– Контекстна діаграма

На рисунку 2.2 представлено діаграму, що показує детальну роботу

торгового радника, що складається з трьох етапів:

- проведення аналізу котирувань та здійснення аналізу;
- вибір торгової стратегії та визначення змінних;
- торгівля.

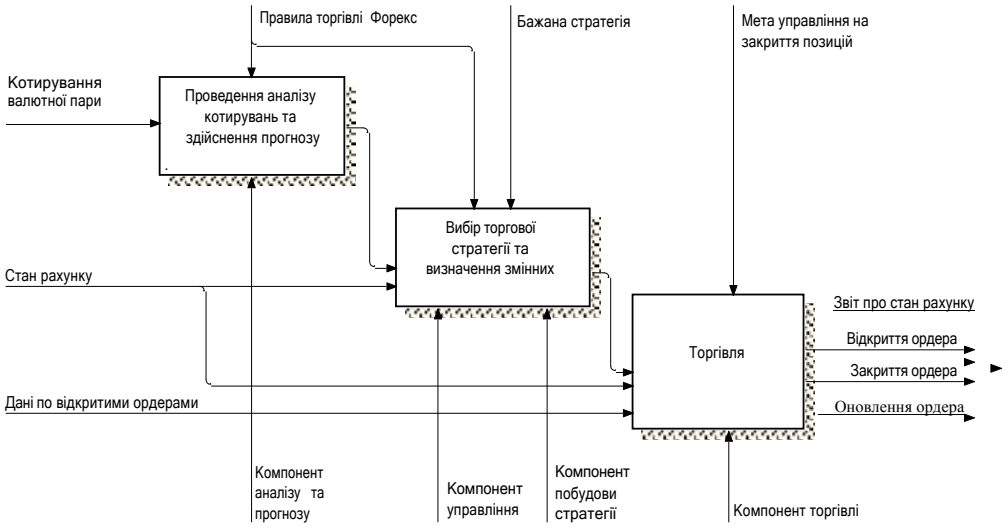

Рисунок 2.2 – Функціональна модель радника

Процес «Проведення аналізу котирувань та здійснення прогнозу» також декомпозується на 4 блоки (див. рис. 2.3):

- отримання історії котирувань;
- аналіз коливань;
- обчислення статистичних коефіцієнтів;
- побудова прогнозу.

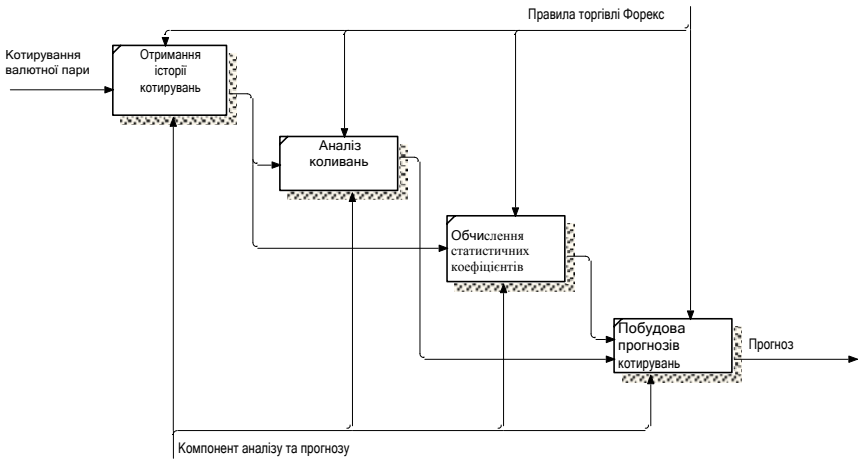

Рисунок 2.3 - Проведення аналізу котирувань та здійснення прогнозу

Вибір торгової стратегії та визначення змінних включає (див. рисунок 2.4):

- визначення торгової стратегії;
- застосування теорії ігор;
- ситуаційні коливання;
- визначення торгових змінних.

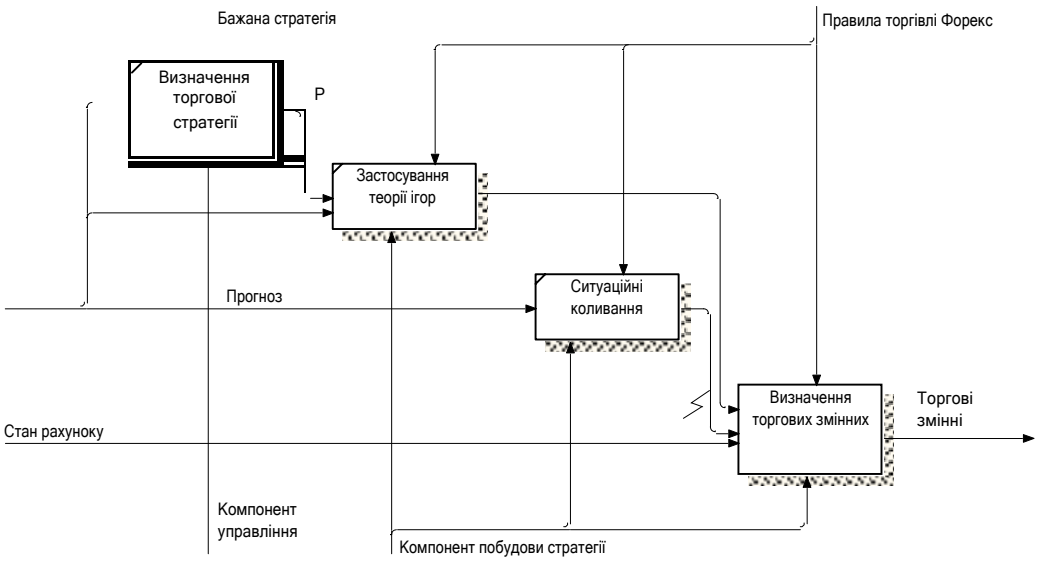

Рисунок 2.4 – Вибір торгової стратегії та визначення змінних

Далі подано процес визначення торгових змінних (див. рис. 2.5).

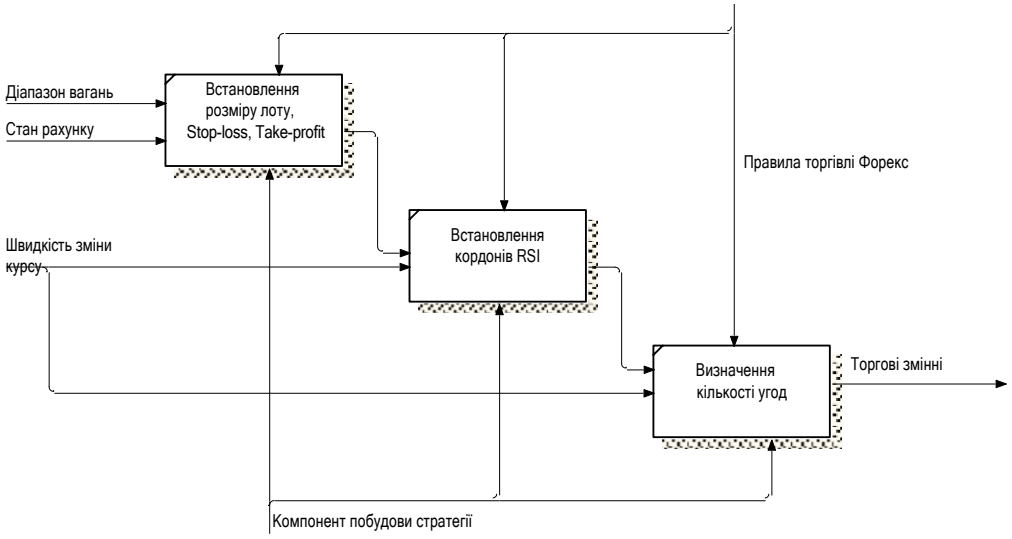

Рисунок 2.5 – Визначення торгових змінних

У структурі радника було виділено 4 основні компоненти, які виконують різні функції:

– компонент аналізу та прогнозу – аналізуючи архів котирувань за попередній період робот здійснює прогноз зміни курсу валют;

– компонент управління – враховуючи розмір цільового прибутку, заданого користувачем, а також очікувану зміну курсу валют та елементи нечіткої логіки, цей компонент визначає, яка з можливих торгових стратегій буде використовуватися надалі;

– компонент побудови стратегії – застосовуючи отриманий прогноз котирувань та результати вирішення типових завдань теорії ігор із середовищем, даний компонент буде задавати торгові змінні для обраної стратегії. Розраховані змінні передаються компоненту торгівлі і є основним інструментом регулювання здійснення ринкових операцій;

– компонент торгівлі - здійснює торгові операції на найпростіший рівень за заздалегідь визначеним змінним.

2.5 Інформаційна модель системи

Інформаційна модель – модель об'єкта, подана у вигляді інформації, що описує суттєві для даного розгляду параметри та змінні величини об'єкта, зв'язки між ними, входи та виходи об'єкта і що дозволяє шляхом подачі на модель інформації про зміни вхідних величин моделювати можливі стани об'єкта. Основна методика побудови інформаційних моделей – DFD.

Метою методики DFD є побудова моделі аналізованої системи як діаграми потоків даних, що забезпечує правильне опис виходів (відгуку системи як даних) при заданому впливі вхід системи (подачі сигналів через зовнішні інтерфейси). Діаграми потоків даних є основним засобом моделювання функціональних вимог до проектованої системи [9].

На рисунку 2.6 наведено діаграму, на якій представлений процес здійснення торгівлі та потоки даних, що показують входи та виходи системи.

Наявність у діаграмах DFD елементів для позначення джерел, приймачів та сховищ даних дозволяє ефективніше та наочно описати процес документообігу

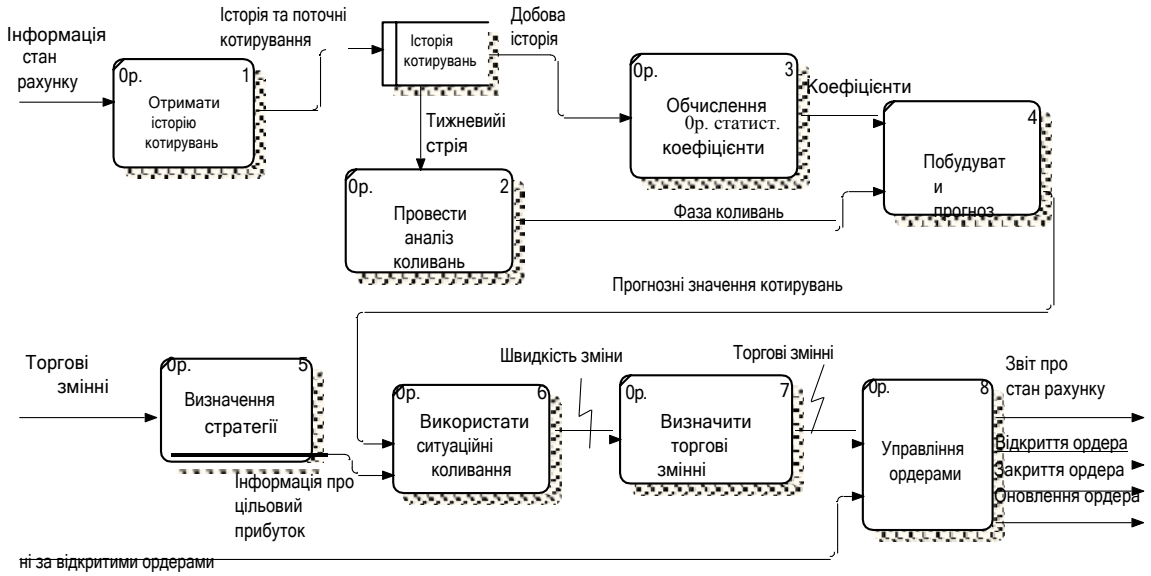

Рисунок 2.6 – Торгівля

2.6 Визначення функцій автоматизованої торгової системи

До основних функцій автоматизованих торгових систем (торгових роботів) відноситься [3]:

- відкриття позиції;
- встановлення ордера take profit;
- встановлення ордера stop loss.

За наявності зазначених функцій радник може торгувати самостійно, відкриваючи та закриваючи позиції.

Крім того, функціонал робота може включати як функції, які служать для підвищення зручності роботи користувачем (висновок інформації про поточний стан роботи програми, відсилання повідомлень користувачу про здійснення операцій по рахунку), так і функції, що підвищують ефективність і прибутковість торгівлі (автоматичне визначення розміру лота, трейлінгу і т.д.). Серед поширених додаткових функцій, що використовуються в раднику, можна навести такі [3]:

- контроль за сумарною позицією;
- перенесення stop loss на рівень беззбитковості;
- трейлінг stop loss;
- закриття позицій за зворотними сигналами індикаторів;
- автоматичне визначення розміру лота;
- повідомлення про відкриття-закриття позицій тощо.

Платформа MetaTrader4 має розвинені можливості для створення та використання автоматичних торгових систем. На мові платформи MetaTrader програма написана мовою MQL4 і відкомпільована засобами MetaEditor називається радником. Для створення, налаштування та використання в торгівлі радників MetaTrader4 має наступні модулі:

- модуль, що містить засоби створення торгових радників (роботів) – MetaEditor
- модуль, який містить засоби для тестування торгових радників –

Тестер Стратегій

MetaQuotes Language 4 (MQL4) – мова програмування торгових радників. Ця мова дозволяє писати власні програми-експерти, які автоматизують управління торговими процесами. Крім того, на MQL4 можна створювати власні технічні індикатори, скрипти та бібліотеки функцій [3].

Мова програмування схожа на C++, а в останніх версіях платформи подібність посилилася після появи можливості об'єктно-

орієнтованого програмування, що дозволяє створювати досить складні системи з можливостями адаптації та учності.

Тестер стратегій– це інструмент торгової платформи MetaTrader4, призначений для здійснення тестування торгових радників, а також для налаштування та оптимізації їх параметрів [2].

Цей модуль дозволяє суттєво прискорити роботу під час перевірки працездатності торгової системи. Тестер стратегій, використовуючи історичні дані, дозволяє переглянути протягом кількох хвилин результати роботи радника кілька років.

Завдяки вбудованим засобам MetaTrader 4 значно спрощується розробка торгового радника, а також його застосування на реальному та демо-рахунку.

2.7 Оптимізація процесів автоматизованої торгівлі

Оптимізація процесів автоматизованої торгівлі спрямована на підвищення ефективності роботи радників через удосконалення алгоритмів, налаштування параметрів, а також використання інноваційних інструментів для аналізу ринку. Зважаючи на динамічність фінансових ринків, торгові стратегії потребують адаптації до змін умов, таких як волатильність, тренди та економічні новини.

Основні напрями оптимізації

1. Удосконалення алгоритмів аналізу

Використання сучасних методів прогнозування, таких як машинне навчання або нейронні мережі, дозволяє радникам швидко аналізувати великі обсяги історичних даних. Це сприяє більш точному прогнозуванню цінових рухів і вибору ефективної торгової стратегії. Наприклад, методи кластерного аналізу можуть бути застосовані для ідентифікації ринкових умов, які повторюються.

2. Оптимізація параметрів радника

- Тестування та налаштування: Використання тестера стратегій для перевірки працездатності радника на історичних даних дозволяє виявити найбільш вигідні параметри.

- Режим оптимізації: Програма MetaTrader 4 має інструмент, що дозволяє автоматизувати підбір параметрів, таких як розмір лота, рівні stop loss і take profit.

- Моніторинг результатів: Постійне спостереження за результатами роботи радника та коригування його параметрів у реальному часі забезпечує стабільність торгівлі.

3. Застосування стратегій ризик-менеджменту

Радники мають бути оснащені функціями, які дозволяють мінімізувати ризики, такі як:

- Обмеження максимальної суми втрат за одну угоду.

- Використання змінного розміру лота залежно від розміру депозиту.

- Впровадження механізмів перенесення ордерів до беззбитковості.

4. Адаптація до ринкових умов

Динамічні ринки вимагають від торгових радників швидкого реагування на зміну трендів, новин та інших зовнішніх факторів. Для цього доцільно використовувати:

- Індикатори, що враховують волатильність (наприклад, ATR – Average True Range).

- Алгоритми, які автоматично перемикаються між різними стратегіями залежно від ринкової ситуації.

5. Використання обчислювальних кластерів

Проведення розрахунків на високопродуктивних обчислювальних платформах значно скорочує час оптимізації параметрів радника, що особливо важливо для складних стратегій з великою кількістю змінних.

Оптимізація процесів автоматизованої торгівлі забезпечує:

- Підвищення прибутковості за рахунок кращого вибору торгових стратегій.

- Зниження ризиків завдяки більш точним налаштуванням параметрів.

- Підвищення надійності роботи радників у реальних умовах.

- Адаптивність до нових ринкових тенденцій.

Таким чином, постійна оптимізація є невід’ємною частиною роботи з автоматизованими торговими системами. Її реалізація дозволяє підвищити ефективність роботи радників, забезпечуючи стабільний дохід навіть у нестабільних ринкових умовах.

3 Створення автоматизованої торгової системи

Редактор MetaEditor є частиною клієнтського терміналу Metatrader 4 і використовується для створення, редагування та компіляції програм, написаних мовою MQL4. Він дозволяє розробляти радники, індикатори, скрипти та бібліотеки функцій. У MetaEditor також інтегрований майстер створення радників, який значно полегшує розробку нових MQL-програм [1].

Для виклику редактора MetaEditor необхідно у вікні терміналу натиснути відповідну клавішу. Після цього відкриється вікно редактора (див. рисунок 3.1).

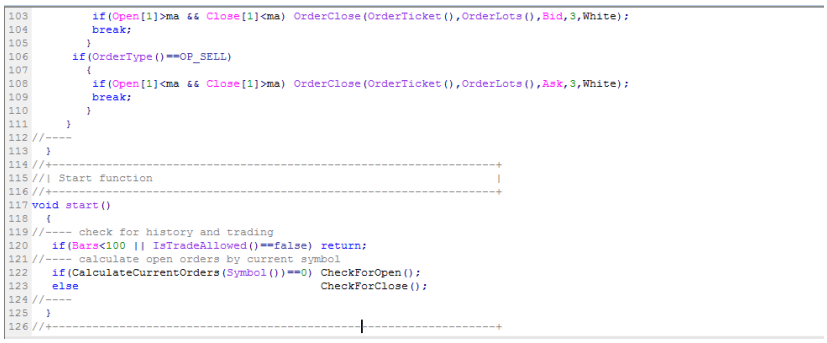

Рисунок 3.1 – Вікно редактора

Далі запустимо майстер створення радника (див. рис.3.2).

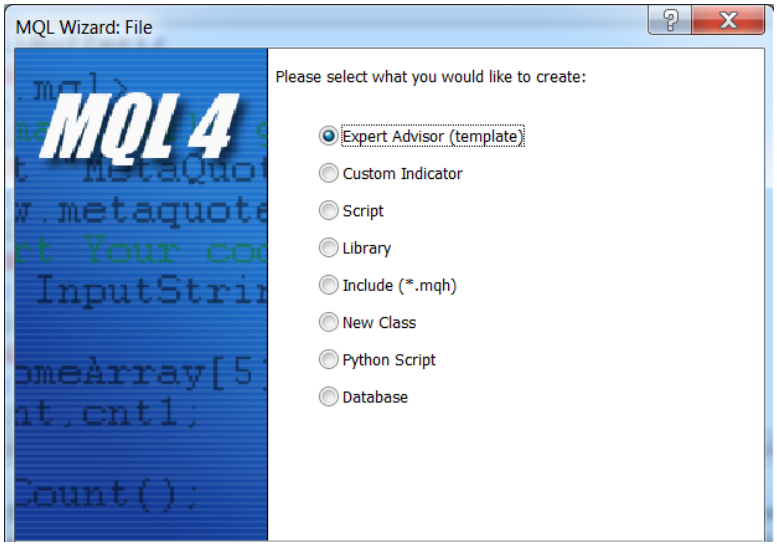

Рисунок 3.2 – Вікно майстра

Майстер «створення радника» (див. рисунок 3.3) пропонує задати

замість нового скрипту, для подальшого його збереження та роботи з ним. Введемо у відповідне поле EMA

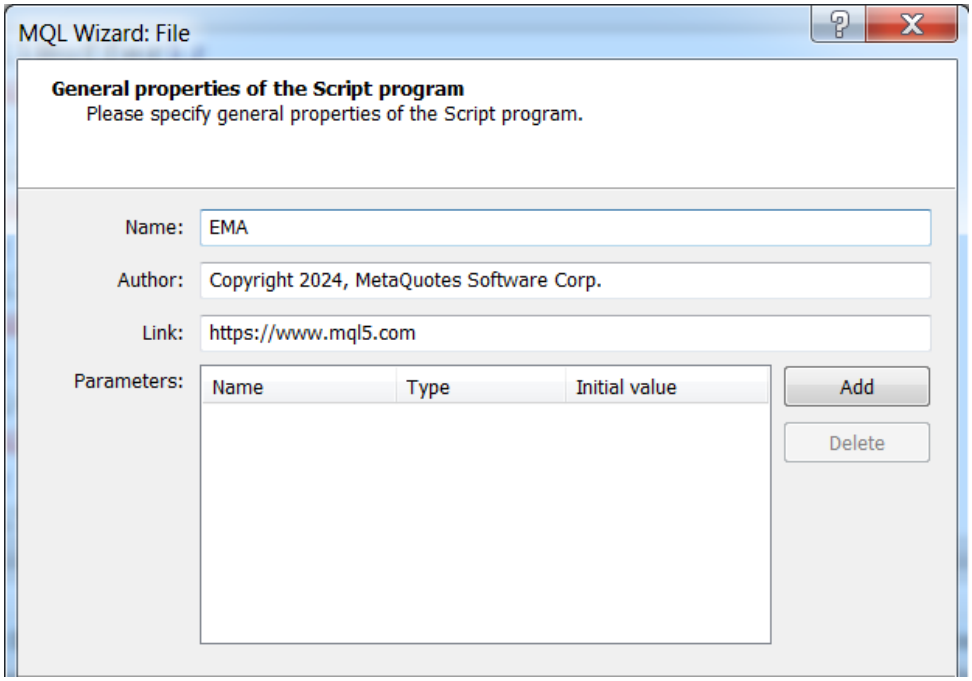

Рисунок 3.3 – Загальні параметри радника

Наступний етап передбачає безпосереднє написання коду радника (рисунок 3.4). Цей процес є ключовим і потребує знання мови MQL4.

```
int start()
{
   if (TimeBar==Time[0]) return(0);
   double EMA0 = iMA(NULL,0,period_EMA,0,MODE_EMA, PRICE_OPEN,0);
   double WMA0 = iMA(NULL,0,period_WMA,0,MODE_LWMA,PRICE_OPEN,0);
   double EMA1 = iMA(NULL,0,period_EMA,0,MODE_EMA, PRICE_OPEN,1);
   double WMA1 = iMA(NULL,0,period_WMA,0,MODE_LWMA,PRICE_OPEN,1);
   double RSI  = iRSI(NULL,0,period_RSI,PRICE_OPEN,0);
   if (EMA0 < WMA0 && EMA1 > WMA1 && RSI > 50)
   {
      TimeBar=Time[0];
      if (takeprofit!=0) TP  = Ask + takeprofit*Point;
      if (stoploss!=0)   SL  = Ask - stoploss*  Point;
      LOT = LOT(risk,1);
      if (CloseCounter) CLOSEORDER("Sell");
      OPENORDER ("Buy");
   }
   if (EMA0 > WMA0 && EMA1 < WMA1 && RSI < 50)
   {
      TimeBar=Time[0];
      if (takeprofit!=0) TP = Bid - takeprofit*Point;
      if (stoploss!=0)   SL = Bid + stoploss*  Point;
      LOT = LOT(risk,1);
      if (CloseCounter) CLOSEORDER("Buy");
```

Рисунок 3.4 – Код радника

3.1 Підготовка до роботи автоматизованої торгової системи

Наступним етапом є підготовка до роботи автоматизованої торгової системи (торгового радника). Можна зустріти два формати, в яких представлено радника:

– як файл з розширенням .ex4;
– як файл з розширенням .mq4.

Готовий до запуску торговий радник – це файл із розширенням .ex4. Він уже готовий до роботи, але не має можливості вносити до цього радника жодних змін.

Файл із розширенням .mq4 – файл, що містить вихідний код торгового радника. Саме у цьому форматі зберігається радник до його остаточного робочого варіанту [2]. Щоб цей варіант радника можна було використовувати у торгівлі, потрібно відкомпілювати файл mq4.

Для того, щоб використовувати в терміналі новий радник, необхідно вчинити наступні дії:

1) Файли із розширенням. ex4 і .mq4 розмістити в папці Experts каталог даних MetaTrader4. За умовчанням це-Terminal MQL4 Experts.

2) Перезапустіть платформу MT4.

3) Увімкнути навігатор

4) Переконайтеся, що у списку радників є доданий (див. рис. 3.5).

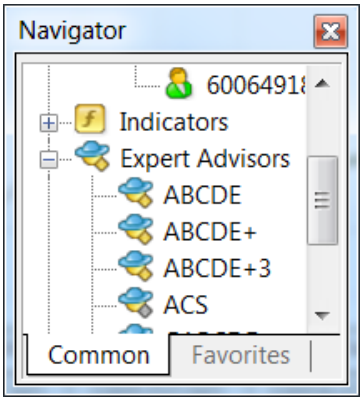

Рисунок 3.5 – Вікно Навігатора

5) Кольорові піктограми у списку позначені радники з доступним вихідним кодом. Вихідний код радника необхідно відкомпілювати.

6) Для компіляції необхідно виділити такий радник, натиснути Enter і у вікні MetaEditor зробити його компіляцію

3.2 Тестування торгового радника

Перед використанням радника на демонстраційному або реальному рахунку необхідно переконатися в його працездатності. Для цього застосовується тестер стратегій із завантаженими історичними даними.

Після того, як у торговій платформі є достатня кількість історичних даних, за допомогою тестера стратегій необхідно запустити створений радник у роботу цих історичних даних. Якщо протягом тривалого часу радник показуватиме хороший результат, значить, є велика ймовірність, що він зможе показати такий самий результат і в майбутньому.

Тестування торгового радника – процедура, що дозволяє перевірити його працездатність із використанням історичних даних [4].

Тестування може проводитися з різним ступенем точності на різних таймфреймах, і охоплювати заданий користувачем період часу.

Після відкриття вікна тестера стратегій, необхідно зробити такі дії:

1) необхідно вибрати потрібний радник зі списку;
2) вибрати інструмент;
3) встановити таймфрейм на якому тестуватиметься радник;
4) Визначити метод тестування.

Далі необхідно налаштувати властивості експерта, натиснувши відповідну кнопку.

Перед початком тестування та оптимізації радника рекомендується встановити значення стандартних параметрів. Для цього натисніть кнопку "Скинути".

Далі вказуємо період для тестування, встановлюючи дати ВІД та ДО, вибираємо модель оптимізації, включаємо/виключаємо режим візуалізації (дана можливість допоможе наочно побачити, яким чином працює торговий радник) та запускаємо тестування кнопкою"Старт"

Результат тестування можна оцінити, використовуючи закладки, що з'явилися: «Результати», «Графік», «Звіт» та «Журнал» [5].

Закладка «Результати» (див. рис. 3.7) – це угоди, проведені радником

на історичних даних. Можливе сортування за зростанням/зменшенням кожної з колонок.

№	Час	Тип	Ордер	Об'єм	Ціна	S/L	T/P	Прибуток	Баланс
1	2024.01.04 00:05	sell	1	0,1	1,4323	0	1,4123		
2	2024.01.04 05:35	sell	2	0,1	1,4287	0	1,4087		
3	2024.01.04 11:30	sell	3	0,1	1,4319	0	1,4182		
4	2024.01.04 15:40	sell	4	0,1	1,442	0	1,4228		
5	2024.01.04 20:40	sell	5	0,1	1,4422	0	1,4222		
6	2024.01.05 01:35	sell	6	0,1	1,4419	0	1,4219		
7	2024.01.05 02:30	sell	7	0,1	1,4418	0	1,4218		
8	2024.01.05 04:45	sell	8	0,1	1,4419	0	1,4219		
9	2024.01.05 06:00	sell	9	0,1	1,4425	0	1,4226		
10	2024.01.05 13:00	sell	10	0,1	1,4417	0	1,4217		
11	2024.01.05 18:50	sell	11	0,1	1,4387	0	1,4187		
12	2024.01.05 22:00	sell	12	0,1	1,4367	0	1,4167		
13	2024.01.06 01:05	sell	13	0,1	1,4362	0	1,4162		
14	2024.01.06 03:20	sell	14	0,1	1,4354	0	1,4154		
15	2024.01.06 07:10	sell	15	0,1	1,4354	0	1,4154		
16	2024.01.06 07:47	modify	4	0,1	1,4428	1,4427	1,4228		
17	2024.01.06 07:47	modify	4	0,1	1,4428	1,4426	1,4228		
18	2024.01.06 07:47	modify	9	0,1	1,4426	1,4426	1,4226		
19	2024.01.06 07:47	modify	4	0,1	1,4428	1,4425	1,4228		
20	2024.01.06 07:47	modify	9	0,1	1,4426	1,4425	1,4226		
21	2024.01.06 07:47	modify	4	0,1	1,4428	1,4424	1,4226		
22	2024.01.06 07:47	modify	9	0,1	1,4426	1,4424	1,4226		
23	2024.01.06 07:47	modify	4	0,1	1,4428	1,4423	1,4228		
24	2024.01.06 07:47	modify	9	0,1	1,4426	1,4423	1,4226		
25	2024.01.06 07:47	modify	4	0,1	1,4428	1,4422	1,4228		
26	2024.01.06 07:47	modify	9	0,1	1,4426	1,4422	1,4226		

Рисунок 3.6 – Результати

Закладка «Графік» (див. рисунок 3.7) – зміна балансу рахунку під час проведення операцій у графічному вигляді.

Горизонтальна шкала – кількість угод. Вертикальна шкала – зміна балансу.

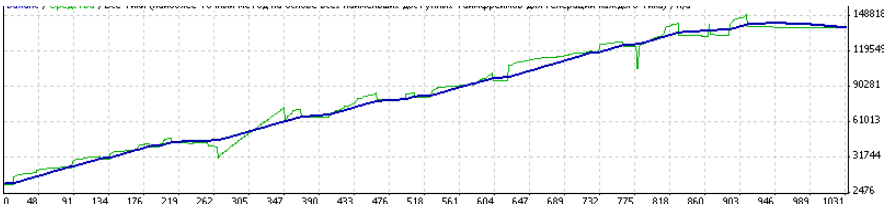

Рисунок 3.7 – Графік

3.3 Оптимізація параметрів торгового радника

Оптимізація параметрів торгового радника є важливим кроком для досягнення його максимальної ефективності на обраному ринку. Оскільки ринки мають різні характеристики, радник, налаштований для одного ринку, може виявитися менш ефективним на іншому. Тому необхідно адаптувати його параметри до специфіки середовища, в якому він буде використовуватися.

Суть оптимізації

Процес оптимізації полягає в підборі таких параметрів, які дозволять враховувати волатильність, тенденції та характер ринкових змін для досягнення максимального позитивного результату. Оптимізація виконується на основі історичних даних, що дає змогу оцінити потенційну ефективність радника. Проте важливо розуміти, що результати, отримані на історичних даних, не гарантують аналогічної ефективності у майбутньому через можливі зміни ринкових умов.

Основні кроки оптимізації

1. Вибір параметрів для оптимізації:

На початку необхідно визначити, які саме параметри радника будуть змінюватися під час оптимізації. Також задається діапазон значень для кожного параметра і крок їх зміни. Це дозволяє дослідити різні комбінації параметрів і вибрати найкращу.

2. Налаштування торгових умов:

Потрібно вказати обсяг початкового депозиту, вибір валюти рахунку, а також додаткові умови, такі як допустимість лише довгих (long) або коротких (short) позицій.

3. Запуск оптимізації:

Оптимізація проводиться за заданий часовий проміжок, протягом якого радник багаторазово тестується з різними комбінаціями параметрів. Це дозволяє оцінити, які налаштування забезпечують найкращі результати.

Оцінка результатів

Після завершення оптимізації результати аналізуються за кількома ключовими показниками:

- Чистий прибуток: Різниця між загальними прибутками та збитками.

- Просідання: Максимальне зниження балансу рахунку за певний період.

- Прибутковість: Відношення загального прибутку до збитків у відсотках.

Ідеальними вважаються параметри, які забезпечують високу прибутковість при мінімальному просіданні. Після аналізу оптимальні налаштування інтегруються у радник.

Методи оптимізації наступні.

Існують два основних підходи:

1. Проста оптимізація:

Проводиться на одному часовому інтервалі, використовуючи історичні дані. Недоліком цього методу є ризик переоптимізації — адаптації параметрів виключно під аналізований період без урахування майбутніх ринкових змін.

2. Оптимізація зі зміщенням вибірки:

Цей метод передбачає виконання оптимізації на одному часовому інтервалі, а потім перевірку отриманих налаштувань на іншому. Це дозволяє уникнути переоптимізації та підвищити надійність результатів.

Оптимізація параметрів є критично важливим процесом, що дозволяє підвищити ефективність радника та адаптувати його до конкретних умов ринку. Використання структурованого підходу та перевірка результатів на різних часових інтервалах мінімізує ризики і покращує надійність автоматизованої торгової системи.

3.4 Перевірка працездатності розробленої торгової системи

Для оцінки функціональності розробленого торгового радника необхідно виконати тестування на демо-рахунку. Це дозволяє перевірити його працездатність у реальних ринкових умовах без ризику втрат реальних

коштів. Процес підключення радника включає кілька основних кроків:

Налаштування інструменту та таймфрейму: Обирається інструмент і таймфрейм, для якого було проведено оптимізацію радника.

Додавання радника на графік:У навігаторі торгової платформи знаходиться потрібний радник, який переноситься на графік обраного інструменту.

Задавання вхідних параметрів:У вікні налаштувань вибирається вкладка "Вхідні параметри", де встановлюються оптимізовані значення параметрів (див. рисунок 3.8)

Налаштування загальних параметрів:У вкладці "Загальні" дозволяється виконання торгівлі радником, використання функцій із зовнішніх бібліотек, а також за необхідності вмикається опція ручного підтвердження угод. Тут також можна обмежити здійснення угод лише для покупок або продажів.

Активація радника.

Для запуску радника необхідно ввімкнути загальну функцію автоторгівлі у торговій платформі, натиснувши відповідну кнопку.

Рисунок 3.8 – Робота радника

Оцінка роботи радника

Працездатність радника підтверджується спеціальним значком у торговій платформі. У процесі роботи можна змінювати його параметри, забороняти окремі види угод або повністю вимкнути. Ці дії виконуються через контекстне меню, доступне при натисканні правою кнопкою миші на значок радника.

Додатково можна відстежувати поточний стан роботи радника у вкладках

терміналу "Експерти" та "Журнал", а також аналізувати лог-файли, що зберігаються у папці Terminal MQL4 logs (див. рисунок 3.9). Лог-файли дозволяють отримати деталізовану інформацію про роботу радника та виявити можливі проблеми або аномалії.

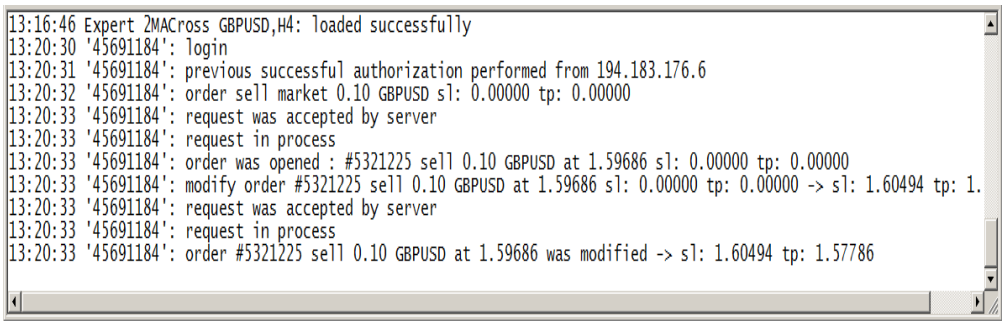

Рисунок 3.9-Результати роботи. Лог файл на оцінка економічної ефективності цієї розробки.

Результати тестування та аналізу роботи радника на демо-рахунку є важливим етапом для оцінки його економічної ефективності. Це дозволяє забезпечити належну підготовку до його використання у реальних ринкових умовах

3.5 Розгортання торгової системи на реальному рахунку

Після успішного тестування та оптимізації торгового радника на демонстраційному рахунку можна перейти до його розгортання на реальному рахунку. Цей етап вимагає ретельної підготовки та перевірки налаштувань, щоб забезпечити стабільну роботу радника в умовах реальної торгівлі.

Кроки розгортання:

Вибір брокера та торгових умов:

Переконайтеся, що брокер підтримує платформу MetaTrader 4 і пропонує сприятливі умови торгівлі, такі як низький спред, швидке виконання ордерів і доступ до необхідних фінансових інструментів.

Налаштування реального рахунку:

Відкрийте рахунок із відповідним початковим депозитом.

Встановіть рівень кредитного плеча, оптимальний для стратегії радника.

Перевірте, чи відповідає тип рахунку вимогам (наприклад, ECN, стандартний або мікрорахунок).

Перенесення радника:

Завантажте радника у папку Experts у каталозі терміналу MetaTrader 4, аналогічно до процесу, описаного в підрозділі 3.1.

Перевірка з’єднання та налаштувань:

Переконайтеся, що термінал підключений до інтернету, а сервер брокера доступний.

Увімкніть автоторгівлю у налаштуваннях платформи.

Встановлення радника на графік:

Додайте радника на графік вибраного фінансового інструменту.

Перевірте правильність оптимізованих параметрів у вкладці «Вхідні параметри».

Активуйте радника, натиснувши кнопку "Автоторгівля" у терміналі.

Моніторинг роботи радника:

Постійно стежте за виконанням угод та станом рахунку.

Використовуйте журнал операцій у платформі для аналізу логів і своєчасного виявлення помилок.

Запобіжні заходи під час роботи:

Обмеження ризиків: Встановіть стоп-лоси та тейк-профіти відповідно до стратегії.

Контроль над параметрами: Регулярно перевіряйте відповідність

поточних умов ринку параметрам, які були оптимізовані раніше.

Резервне копіювання: Зберігайте резервні копії радника та налаштувань у безпечному місці, щоб уникнути втрати даних у разі технічних збоїв.

Висновок

У рамках дослідження було підтверджено переваги використання торгових роботів у порівнянні з ручною торгівлею. Разом із цим встановлено, що існуючі торгові радники часто є або недостатньо ефективними, або мають обмежену сферу застосування. На основі обраних методів технічного аналізу визначено концептуальну базу для створення нового радника. Проведені випробування популярних рішень виявили їхні слабкі сторони, що стало основою для виділення ключових компонентів: аналізу та прогнозування, формування стратегії, управління та торгівлі.

У ході дослідження також було розглянуто перспективні підходи до прогнозування та прийняття рішень, такі як нейронні мережі, нечітка логіка та зворотний зв'язок для адаптивного перерахунку стратегії. Наступним етапом є розробка математичного апарату, програмування радника та подальше його тестування.

Досягнуті результати:

Проведено аналіз методів зниження помилок прогнозу та мінімізації збиткових угод.

Визначено функціональні та експлуатаційні вимоги до автоматизованої інформаційної системи.

Досліджено програмні інструменти для створення торгових радників.

Розглянуто існуючі методи виконання торгових операцій, прогнозування валютних курсів та побудови стратегій.

Оцінено характеристики і можливості існуючих методів прогнозування для автоматичної торгівлі.

Проаналізовано сучасні технології торгівлі.

Сформовано структуру майбутньої торгової системи, з деталізацією її компонентів та їх взаємодії.

Проведено тестування створеного торгового радника.

Результатом дослідження є створення автоматизованої торгової системи для роботи на ринку Forex. Система має низку важливих переваг, серед яких:

- ефективне виконання торгових операцій із мінімальними втратами;
- висока швидкість аналізу та обробки даних;
- можливість вирішення нових завдань і адаптації до змін ринку.

Подальші перспективи

Розроблена система може бути розширена за рахунок інтеграції кількох торгових стратегій, що підвищить ймовірність отримання прибуткових угод. Надалі можливе доопрацювання системи для врахування більш складних ринкових умов і розширення функціональних можливостей.

Список використаних джерел

1) Вільямс, Л. Торгові стратегії на основі технічного аналізу [Текст] / Л. Вільямс. – Київ: Фінанси та аналітика, 2021. – 200 с.

2) Елдер, А. Як грати та вигравати на біржі [Текст] / А. Елдер. – Львів: Астролябія, 2019. – 350 с.

3) Швагер, Д. Технічний аналіз: Повний курс [Текст] / Д. Швагер. – Київ: Класика трейдингу, 2020. – 300 с.

4) Насім Талеб, Н. Чорний лебідь. Вплив малоймовірного [Текст] / Н. Насім Талеб. – Харків: Віват, 2018. – 400 с.

5) Коваль, О. Автоматизація торгових стратегій: теорія та практика [Текст] / О. Коваль. – Львів: Видавництво «Інновації», 2020. – 280 с.

6) Базилевський, І. Основи біржової торгівлі [Текст] / І. Базилевський. – Одеса: Економіка і менеджмент, 2017. – 220 с.

7) Прайс, Д. Алгоритмічна торгівля: введення в автоматизацію [Текст] / Д. Прайс. – Київ: Фондова аналітика, 2021. – 260 с.

8) Маркет, С. Технічний аналіз фінансових ринків [Текст] / С. Маркет. – Івано-Франківськ: Академія трейдингу, 2020. – 320 с.

9) Петрюк, А. Методи прогнозування фінансових ринків [Текст] / А. Петрюк. – Київ: Наукова думка, 2019. – 180 с.

10) Власенко, Ю. Використання штучного інтелекту у фінансовій аналітиці [Текст] / Ю. Власенко. – Харків: Основа, 2020. – 240 с.

Додаток

```
//+------------------------------------------------------------------+
//|                                                      EMA.mq4 |
//|                  Copyright 2024, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property strict

input  double v8=8;
input  double v15=15;
input  double v28=28;
input  double v30=30;
input  int v39=39;
input  double StopLost=300;

string  SIM1="EURUSD";
string  SIM2="EURGBP";
string  SIM3="EURAUD";
string  SIM4="EURCHF";
string  SIM5="GBPUSD";
string  SIM6="GBPAUD";
string  SIM7="GBPCHF";
string  SIM8="AUDUSD";
string  SIM9="AUDCHF";
string  SIM10="USDCHF";

string  VAL1="EUR";
string  VAL2="GBP";
string  VAL3="AUD";
string  VAL4="USD";
string  VAL5="CHF";

string SIM[20];
string VAL[10];
int inp[20];
int inpic[20];
double Aib[20];
int vector[10];
int tractor[10];
int trend[20];
int indext[20];

double profit[20];

int vectors[10];
double Sever[20];
double forlimit[20];

void OnTick()
  {
  SIM[1]=SIM1;SIM[2]=SIM2;SIM[3]=SIM3;SIM[4]=SIM4;SIM[5]=SIM5;
```

```
  SIM[6]=SIM6;SIM[7]=SIM7;SIM[8]=SIM8;SIM[9]=SIM9;SIM[10]=SIM10;
  VAL[1]=VAL1;VAL[2]=VAL2;VAL[3]=VAL3;VAL[4]=VAL4;VAL[5]=VAL5;
//+------------------------------------------------------------------+
  int total=Logall(1);
  int times=End();

  if (times==1 && total==0)times=0;
  double histor=Histor();
  if (histor>(v8*v15)&&OrdersTotal()>0&&total==0)
   {total=OrdersTotal();
   while (total>0){DELALL();total=OrdersTotal();}
   Print("histor= ",histor);
   return;}
  if (histor>(v8*v15)&&OrdersTotal()==0)return;
  if (Hour()<3&&OrdersTotal()==0)return;
  double pribel=Pribel(),Plus=Plus(),Minus=Minus();
  double MPlus=MPlus(),MMinus=MMinus();
//+------------------------------------------------------------------+
  ArrayInitialize(trend,0);
  for(int i=1;i<=10;i++){EXT(SIM[i],i);}
//+------------------------------------------------------------------+
  ArrayInitialize(vector,0);
   for(int i=1;i<=5;i++)
    {vector[i]=0;
     for(int j=1;j<=10;j++)
     {
     if (StringFind(SIM[j],VAL[i],0)==0)
         {if(inp[j]>0)vector[i]++;if(inp[j]<0)vector[i]--;}
     if (StringFind(SIM[j],VAL[i],0)==3)
         {if(inp[j]>0)vector[i]--;if(inp[j]<0)vector[i]++;}
     }
    }
//+------------------------------------------------------------------+
 ArrayInitialize(tractor,0);
   for(int i=1;i<=5;i++)
    {tractor[i]=0;
     for(int j=1;j<=10;j++)
     {
     if (StringFind(SIM[j],VAL[i],0)==0)
         {if(trend[j]>0)tractor[i]++;if(trend[j]<0)tractor[i]--;}
     if (StringFind(SIM[j],VAL[i],0)==3)
         {if(trend[j]>0)tractor[i]--;if(trend[j]<0)tractor[i]++;}
     }
    }
//+------------------------------------------------------------------+
  int Kplus=Kplus();int Kminus=Kminus();
  double u=1;
  if (Kminus>=Kplus)u=-1;
//+------------------goot ---------------------------------------------+
  int goot=0;if(Standart()==5)goot=1;
//+---------- ------------------------------------------------------+
  if (total>0)
  {
  double v0=v();
```

```
  double LIN=MathMax(1, total);
  if (Minus>Plus&&times==0)LIN=MathMax(1, v8-Kminus);

   if (Plus+Minus>(v0-Kminus)*LIN)
         {while (total>0){DELALL();total=OrdersTotal();}return;}

   if (pribel>=v8&&histor>=(v8*v15))
         {while (total>0){DELALL();total=OrdersTotal();}return;}

  if (MPlus/MathAbs(MMinus)>=(1+u*double(v8)/100)&&
MPlus/MathAbs(MMinus)<=(1+double(v28)/100))
         {while (total>0){DELALL();total=OrdersTotal();}return;}
  }
//+------------------------------------------------------------------+
   for(int j=1;j<=10;j++)
     {string SIMic=SIM[j];
     int Limutfase=Logsim(SIMic,2);int Openfase=Logsim(SIMic,1);
     if (Openfase>=2){DELTICK(SIMic);break;}
 total=Logall(1);
//+------------------------------------------------------------------+
      for (int i=0;i<OrdersTotal(); i++)
     {if(OrderSelect(i, SELECT_BY_POS)==true &&OrderSymbol()==SIMic)
        {
        double Openprice=0;double point=MarketInfo(SIMic,MODE_POINT);
        if (point==0.00001)point=0.0001;if (point==0.001)point=0.01;
        double ask=MarketInfo(SIMic,MODE_ASK);double bid=MarketInfo(SIMic,MODE_BID);
        double spread=(ask-bid)/point;
        if (spread>=3) spread=3;
         else spread=2;

        if (Limutfase>0){
        if(OrderType()==OP_BUYLIMIT)Openprice=OrderStopLoss()+StopLost*point;
        if(OrderType()==OP_SELLLIMIT)Openprice=OrderStopLoss()-StopLost*point;

        if(OrderType()==OP_BUYLIMIT&&trend[j]==3&& (Openprice-
ask)/point>=profit[j]&&Openfase>0)
        {DELBUYLIMIT(SIMic);MODI3(SIMic,j);break;}
        if(OrderType()==OP_SELLLIMIT&&trend[j]==-3&& (bid-
Openprice)/point>=profit[j]&&Openfase>0)
        {DELSELLLIMIT(SIMic);MODI3(SIMic,j);break;}

        if(OrderType()==OP_BUYLIMIT&&trend[j]==-100&&Openfase==0&&goot==1)
        {if (SELLLIMIT(SIMic,j)==2)break;}
        if(OrderType()==OP_SELLLIMIT&&trend[j]==100&&Openfase==0&&goot==1)
        {if (BUYLIMIT(SIMic,j)==2)break;}

        if(OrderType()==OP_BUYLIMIT&& (ask-Openprice)/point>=spread+1&&trend[j]<0)
        {DELLIMIT(SIMic);BUY(SIMic,j,0);break;}
        if(OrderType()==OP_SELLLIMIT&& (Openprice-bid)/point>=spread+1&&trend[j]>0)
        {DELLIMIT(SIMic);SELL(SIMic,j,0);break;}

        if(OrderType()==OP_BUYLIMIT&&(Openprice-ask)/point>=spread&&trend[j]>0)
                  {MODILIMIT(SIMic,j);break;}
        if(OrderType()==OP_SELLLIMIT&&(bid-Openprice)/point>=spread&&trend[j]<0)
```

```
                {MODILIMIT(SIMic,j);break;}
        }

        if (Limutfase==0&&Openfase>0)if(OrderTakeProfit()==0)MODI3(SIMic,j);

        if (Limutfase==0&&Openfase>0&&goot==10&&total<7){
        if(OrderType()==OP_BUY&&trend[j]==-10)
        SELLLIMIT2(SIMic,j);
        if(OrderType()==OP_SELL&&trend[j]==10)
        BUYLIMIT2(SIMic,j);}
        }
        }
        }
total=Logall(1);
if (total>=7&&Logall(2)>0){for(int j=1;j<=10;j++)
   {int v=0;for (int i=0;i<OrdersTotal(); i++)
      {if(OrderSelect(i, SELECT_BY_POS)==true &&OrderSymbol()==SIM[j])
      if(OrderType()==OP_BUY||OrderType()==OP_SELL)v=1;}
   if (v==0)if (DELLIMIT(SIM[j])==1)Print(" Лишнее");}}
if (total>=7&&Logall(2)>0){for(int j=1;j<=10;j++)
   {if (DELLIMIT(SIM[j])==1)Print(" Лишнее2");}}

//+------------------------------------------------------------------+
if (total<7&&times!=1)
{int fs=0;
     for(int j=1;j<=10;j++)
     {if(trend[j]==10)if (BUYLIMIT(SIM[j],j)==2){trend[j]=0;fs=1;}
     if(trend[j]==-10)if (SELLLIMIT(SIM[j],j)==2){trend[j]=0;fs=1;}
      }
 }
 }
//------------------------------------------------------------+

int Sessions(int ho)
{
int goot=0;
int AsiaBegin    =  02 ;   // Opening of the Asian session
int AsiaEnd      =  12 ;   // Asian session close
int EurBegin     =  06 ;   // Opening of the European session
int EurEnd       =  17 ;   // Closing of the European session
int USABegin     =  15 ;   // Opening of the American session
int USAEnd       =  23 ;   // Closing of the American session

if (ho>=AsiaBegin&&ho<AsiaEnd)goot=1;
if (ho>=EurBegin&&ho<EurBegin)goot=1;
if (ho>=USABegin&&ho<USAEnd)goot=1;

return(goot);
}
int Standart()
{
 ArrayInitialize(vectors,0);
 ArrayCopy(vectors,vector,0,0,WHOLE_ARRAY);
 int s=0;
```

```
 int index0=0;
 for(int i=1;i<=5;i++) if (MathAbs(vectors[i])==0) index0=i;
 if (index0!=0)s++;

 int indexmax2=0; int indexmin2=0;
 for(int i=1;i<=5;i++) if (MathAbs(vectors[i])==2) indexmax2=i;
 if (indexmax2!=0)s++;

 for(int i=1;i<=5;i++) if (MathAbs(vectors[i])==2 && i!=indexmax2) indexmin2=i;
 if (indexmin2!=0)s++;

 int indexmax4=0; int indexmin4=0;
 for(int i=1;i<=5;i++) if (MathAbs(vectors[i])==4) indexmax4=i;
 if (indexmax4!=0)s++;

 for(int i=1;i<=5;i++) if (MathAbs(vectors[i])==4 && i!=indexmax4) indexmin4=i;
 if (indexmin4!=0)s++;
 ArrayInitialize(vectors,0);

 if (s==5){for(int j=1;j<=10;j++)
   {int index1=0,index2=0;
    if (MathAbs(trend[j])==3)
     {
      for(int i=1;i<=5;i++)
       {
        if(StringSubstr(SIM[j],0,3)==VAL[i])index1=i;
        if(StringSubstr(SIM[j],3,3)==VAL[i])index2=i;
       }

        for(int i=1;i<=10;i++)if (MathAbs(trend[i])==1)Proverka(i,index1,index2);

        }
     }}
return(s);
}
int Proverka(int j,int indexmax,int indexmin)
{
  string Valutamax4=VAL[indexmax];
  string Valutamin4=VAL[indexmin];
  if ((StringFind(SIM[j],Valutamax4,0)==0||StringFind(SIM[j],Valutamin4,0)==0)
     && MathAbs(trend[j])==1){if (trend[j]>0)trend[j]=10;else trend[j]=-10;}
  if ((StringFind(SIM[j],Valutamax4,0)==3||StringFind(SIM[j],Valutamin4,0)==3)
     && MathAbs(trend[j])==1){if (trend[j]>0)trend[j]=10;else trend[j]=-10;}
 return(0);
}

double EXT(string SIMic, int j)
{
  double point=MarketInfo(SIMic,MODE_POINT);int okr=4;
  if (point==0.00001)point=0.0001;
  if (point==0.001)point=0.01;
  if (point==0.01)okr=2;
  double ask=MarketInfo(SIMic,MODE_ASK);
  double bid=MarketInfo(SIMic,MODE_BID);
```

```
   double spread=(ask-bid)/point;
   spread=MathMax(spread,2);
   double help=Nord(SIMic,v28,v8);
   Aib[j]=help;

   if (spread>=3) spread=3; else spread=2;

   datetime some_time=iTime(SIMic,PERIOD_W1,0);
   int shift=iBarShift(SIMic,0,some_time,false);

   some_time=iTime(SIMic,PERIOD_D1,0);
   int      shift0=iBarShift(SIMic,0,some_time,false);

   double max=iLow(SIMic,0,shift);
   double min=iHigh(SIMic,0,shift);

   int zero=(int)v8;

   int vektor=0;
   int buy=0,sell=0;
   int index=shift;
   int im30=shift;

   if (spread>=3) spread=3; else spread=2;

for(int i=shift;i>=1;i--)
   {
   if (vektor>=0&&(iHigh(SIMic,0,i)-iHigh(SIMic,0,index))/point>0)
     {buy++;sell=0;zero=buy;
     vektor=1;index=i;
     max=iLow(SIMic,0,i);}//+-Red +

   else if (vektor<=0&&(iLow(SIMic,0,index)-iLow(SIMic,0,i))/point>0)
     {sell++;buy=0;zero=sell;
     vektor=-1;index=i;
     min=iHigh(SIMic,0,i);}//-Blue

   else if (vektor<=0&&(iLow(SIMic,0,i)-min)/point>=zero)
     {max=iLow(SIMic,0,i);zero=-1;
     buy++;sell=0;
     if (buy==1)im30=i;
     vektor=2;index=i;}

   else if (vektor>=0&&(max-iHigh(SIMic,0,i))/point>=zero)
     {min=iHigh(SIMic,0,i);zero=-1;
     sell++;buy=0;
     if (sell==1)im30=i;
     vektor=-2;index=i;}

   else zero--;
   }
   double Newspread=Nord(SIMic,v8,spread+1);
   profit[j]=MathMax(MathAbs(v8-Newspread),spread+1);
   if (buy>0) trend[j]=-3;
```

```
   if (sell>0) trend[j]=3;
   if (buy==1&&im30<=2) trend[j]=-1;
   if (sell==1&&im30<=2) trend[j]=1;

   Sever[j]=MathMax(v8,spread);

   inp[j]=trend[j];

   forlimit[j]=StopLost;

  return(trend[j]);
}
int Wnord(string SIMic, int ot, int doo)
{
long s=0;long w=0;
for(int i=ot;i<doo;i++)
    {w+=iVolume(SIMic,0,i);s++;}
if(s!=0)w=w/s;
int Sw=(int)w/10;
return(Sw);}

double Nord(string SIMic, double v, double min)
{

double point=MarketInfo(SIMic,MODE_POINT);int okr=4;
if (point==0.00001)point=0.0001;
if (point==0.001)point=0.01;
if (point==0.01)okr=2;
double ask=MarketInfo(SIMic,MODE_ASK);
double bid=MarketInfo(SIMic,MODE_BID);

double moda=0;
double s=0;

for(int i=1;i<10;i++)
   {
   double highi=MathMax(iHigh(SIM[i],PERIOD_H1,2),
            MathMax(iHigh(SIM[i],PERIOD_H1,0),iHigh(SIM[i],PERIOD_H1,1)));
   double lowi=MathMin(iLow(SIM[i],PERIOD_H1,2),
            MathMin(iLow(SIM[i],PERIOD_H1,0),iLow(SIM[i],PERIOD_H1,1)));
   if ((highi-lowi)!=0){moda+=1/((highi-lowi)/point);s++;}}

   if (s!=0)moda=moda/s;

   double highi=MathMax(iHigh(SIMic,PERIOD_H1,2),
            MathMax(iHigh(SIMic,PERIOD_H1,0),iHigh(SIMic,PERIOD_H1,1)));
   double lowi=MathMin(iLow(SIMic,PERIOD_H1,2),
            MathMin(iLow(SIMic,PERIOD_H1,0),iLow(SIMic,PERIOD_H1,1)));

   double k=1;
   if (moda!=0)k=(1/((highi-lowi)/point)/moda);
   double help=(int)MathMax(min,v*k);

return(help);
```

```
}
int BUY(string SIMic, int j, double lt)
{
if (Sessions(Hour())==0)return(0);
double point=MarketInfo(SIMic,MODE_POINT);int okr=4;
if (point==0.00001)point=0.0001;
if (point==0.001)point=0.01;
if (point==0.01)okr=2;
double ask=MarketInfo(SIMic,MODE_ASK);
double bid=MarketInfo(SIMic,MODE_BID);
double take=NormalizeDouble(ask+(Aib[j]+profit[j])*point,okr);
     double spread=(ask-bid)/point;
  if (spread>5)return(0);
int total=OrdersTotal();
for (int i=0;i<=total;i++)
  {if(OrderSelect(i, SELECT_BY_POS)==true &&OrderSymbol()==SIMic)
       if(OrderType()==OP_BUYLIMIT||OrderType()==OP_BUY)return(0);}
double Lots=Lot(SIMic);
if (lt>0)Lots=lt*Lots;

int ticket=OrderSend(SIMic,OP_BUY,Lots,ask,3,0,take);
if(ticket<0){Print(SIMic," ask= ",ask, "Error opening BUY order : ",GetLastError());return(-1);}
return(2);
}
int SELL(string SIMic, int j,double lt)
{
if (Sessions(Hour())==0)return(0);
double point=MarketInfo(SIMic,MODE_POINT);int okr=4;
if (point==0.00001)point=0.0001;
if (point==0.001)point=0.01;
if (point==0.01)okr=2;
double ask=MarketInfo(SIMic,MODE_ASK);
double bid=MarketInfo(SIMic,MODE_BID);
double take=NormalizeDouble(bid-(Aib[j]+profit[j])*point,okr);
  double spread=(ask-bid)/point;
  if (spread>5)return(0);
int total=OrdersTotal();
for (int i=0;i<=total;i++)
  {if(OrderSelect(i, SELECT_BY_POS)==true &&OrderSymbol()==SIMic)
       if(OrderType()==OP_SELLLIMIT||OrderType()==OP_SELL)return(0);}
double Lots=Lot(SIMic);
if (lt>0)Lots=lt*Lots;

int ticket=OrderSend(SIMic,OP_SELL,Lots,bid,3,0,take);
if(ticket<0){Print(SIMic," bid= ",bid," Error opening SELL order : ",GetLastError());return(-1);}
return(2);
}
int BUYLIMIT(string SIMic, int j)
{
if (Sessions(Hour())==0)return(0);
double point=MarketInfo(SIMic,MODE_POINT);int okr=4;
if (point==0.00001)point=0.0001;
if (point==0.001)point=0.01;
if (point==0.01)okr=2;
```

```
double ask=MarketInfo(SIMic,MODE_ASK);
  double bid=MarketInfo(SIMic,MODE_BID);
  double spread=(ask-bid)/point;
  if (spread>5)return(0);
int total=OrdersTotal();
for (int i=0;i<=total;i++)
  {if(OrderSelect(i, SELECT_BY_POS)==true &&OrderSymbol()==SIMic)

if(OrderType()==OP_BUYLIMIT||OrderType()==OP_BUY||OrderType()==OP_SELL)return(0);}
double Lots=Lot(SIMic);

double price=NormalizeDouble(ask-forlimit[j]*point,okr);
double stoploss=NormalizeDouble(ask-StopLost*point,okr);
int ticket=OrderSend(SIMic,OP_BUYLIMIT,Lots,price,3,stoploss,0);
if(ticket<0){Print(SIMic," ask= ",ask," price= ",price, "Error opening BUYLIMIT order :
",GetLastError());return(-1);}
return(2);
}
int SELLLIMIT(string SIMic, int j)
{
if (Sessions(Hour())==0)return(0);
double point=MarketInfo(SIMic,MODE_POINT);int okr=4;
if (point==0.00001)point=0.0001;
if (point==0.001)point=0.01;
if (point==0.01)okr=2;
double ask=MarketInfo(SIMic,MODE_ASK);
  double bid=MarketInfo(SIMic,MODE_BID);
  double spread=(ask-bid)/point;
  if (spread>5)return(0);
int total=OrdersTotal();
for (int i=0;i<=total;i++)
  {if(OrderSelect(i, SELECT_BY_POS)==true &&OrderSymbol()==SIMic)

if(OrderType()==OP_SELLLIMIT||OrderType()==OP_SELL||OrderType()==OP_BUY)return(0);}
double Lots=Lot(SIMic);
double price=NormalizeDouble(bid+forlimit[j]*point,okr);
double stoploss=NormalizeDouble(bid+StopLost*point,okr);
int ticket=OrderSend(SIMic,OP_SELLLIMIT,Lots,price,3,stoploss,0);
if(ticket<0){Print(SIMic," bid= ",bid," price= ",price," Error opening SELLLIMIT order :
",GetLastError());return(-1);}
return(2);
}
int BUYLIMIT2(string SIMic, int j)
{
if (Sessions(Hour())==0)return(0);
double point=MarketInfo(SIMic,MODE_POINT);int okr=4;
if (point==0.00001)point=0.0001;
if (point==0.001)point=0.01;
if (point==0.01)okr=2;
int total=OrdersTotal();
double ask=MarketInfo(SIMic,MODE_ASK);
double price=NormalizeDouble(ask-forlimit[j]*point,okr);
for (int i=0;i<=total;i++)
if(OrderSelect(i, SELECT_BY_POS)==true &&OrderSymbol()==SIMic)
```

```
  if(OrderType()==OP_BUYLIMIT||OrderType()==OP_BUY)return(0);

for (int i=0;i<=total;i++)
if(OrderSelect(i, SELECT_BY_POS)==true &&OrderSymbol()==SIMic)
        {if(OrderType()==OP_SELL)
         {price=NormalizeDouble(ask-forlimit[j]*point,okr);
         if (price>OrderOpenPrice())price=NormalizeDouble(OrderOpenPrice()-forlimit[j]*point,okr);
         if(!OrderModify(OrderTicket(),OrderOpenPrice(),OrderStopLoss(),price,0,0))
         Print("Order modification error. Error code=",GetLastError());
         else Print("order modifications OP_SELL");}}

double Lots=Lot(SIMic);
double stoploss=NormalizeDouble(ask-StopLost*point,okr);
int ticket=OrderSend(SIMic,OP_BUYLIMIT,Lots,price,3,stoploss,0);
if(ticket<0){Print(SIMic," ask= ",ask," stoploss= ",stoploss, "Error opening BUYLIMIT2 order :
",GetLastError());return(-1);}
return(2);
}
int SELLLIMIT2(string SIMic, int j)
{
if (Sessions(Hour())==0)return(0);
double point=MarketInfo(SIMic,MODE_POINT);int okr=4;
if (point==0.00001)point=0.0001;
if (point==0.001)point=0.01;
if (point==0.01)okr=2;
int total=OrdersTotal();
double bid=MarketInfo(SIMic,MODE_BID);
double price=NormalizeDouble(bid+forlimit[j]*point,okr);
for (int i=0;i<=total;i++)
if(OrderSelect(i, SELECT_BY_POS)==true &&OrderSymbol()==SIMic)
       if(OrderType()==OP_SELLLIMIT||OrderType()==OP_SELL)return(0);
for (int i=0;i<=total;i++)
if(OrderSelect(i, SELECT_BY_POS)==true &&OrderSymbol()==SIMic)
        {if(OrderType()==OP_BUY)
         {price=NormalizeDouble(bid+forlimit[j]*point,okr);
         if (price<OrderOpenPrice())price=NormalizeDouble(OrderOpenPrice()+forlimit[j]*point,okr);
         if(!OrderModify(OrderTicket(),OrderOpenPrice(),OrderStopLoss(),price,0,0))
         Print("Order modification error. Error code=",GetLastError());
         else Print("order modificationsOP_BUY");}}
double Lots=Lot(SIMic);
double stoploss=NormalizeDouble(bid+StopLost*point,okr);
int ticket=OrderSend(SIMic,OP_SELLLIMIT,Lots,price,3,stoploss,0);
if(ticket<0){Print(SIMic," bid= ",bid," stoploss= ",stoploss," Error opening SELLLIMIT2 order :
",GetLastError());return(-1);}
return(2);
}
double Lot(string SIMic)
{
double Margin=AccountBalance()/1000*0.2/3;
double kf=1;
string Valuta=AccountCurrency();
string Val0="";
string SimS="";
if (StringFind(SIMic,Valuta,0)!=0&&StringFind(SIMic,Valuta,0)!=3)
```

```
   {
    for(int i=1;i<=5;i++)if (StringFind(SIMic,VAL[i],0)==0)Val0=VAL[i];
    for(int j=1;j<=10;j++)if (StringFind(SIM[j],Val0,0)==0
                          &&StringFind(SIM[j],Valuta,0)==3)SimS=SIM[j];
    if (SimS!="")kf=iClose(SIMic,0,0)/iClose(SimS,0,0);
   }
  if (StringFind(SIMic,Valuta,0)==0) kf=iClose(SIMic,0,0);
double lot=Margin;
if (kf!=0)lot=NormalizeDouble(Margin*kf,2);
if (lot<=0.01)lot=0.01;
return(lot);
}
double Pribel()
{
int total=OrdersTotal();string symbol="";
double point=0.0001;double Spribl=0;
for (int i=0;i<total; i++)
  {if(OrderSelect(i, SELECT_BY_POS)==true)
   symbol=OrderSymbol();
   double bid=MarketInfo(symbol,MODE_BID);
   double ask=MarketInfo(symbol,MODE_ASK);
   double spread=-(ask-bid)/point;
   double Price=OrderOpenPrice();
   double sp=0;
   if (OrderType()==OP_BUY){sp=(bid-Price)/point;Spribl+=sp;}
   if (OrderType()==OP_SELL){sp=(Price-ask)/point;Spribl+=sp;}
   }
return(Spribl);
}
double MPlus()

{
int total=OrdersTotal();string symbol="";
double point=0.0001;double Spribl=v28;
for (int i=0;i<total; i++)
  {if(OrderSelect(i, SELECT_BY_POS)==true)
   symbol=OrderSymbol();
   double bid=MarketInfo(symbol,MODE_BID);
   double ask=MarketInfo(symbol,MODE_ASK);
   point=MarketInfo(symbol,MODE_POINT);
   double spread=MarketInfo(symbol,MODE_SPREAD);
   if (point==0.00001){point=0.0001;spread=spread*0.1;}
   if (point==0.001){point=0.01;spread=spread*0.1;}
   double Price=OrderOpenPrice();
   double sp=0;
   if (OrderType()==OP_BUY){sp=(bid-Price)/point;Spribl=MathMax(sp,Spribl);}
   if (OrderType()==OP_SELL){sp=(Price-ask)/point;Spribl=MathMax(sp,Spribl);}
   }
 if (Spribl==v28)return(0);
return(Spribl);
}
double Plus()
{
int total=OrdersTotal();string symbol="";
```

```
double point=0.0001;double Spribl=0;
int m=0;
for (int i=0;i<total; i++)
   {if(OrderSelect(i, SELECT_BY_POS)==true)
    symbol=OrderSymbol();
    for(int j=1;j<=10;j++)if (symbol==SIM[j])m=j;
    double bid=MarketInfo(symbol,MODE_BID);
    double ask=MarketInfo(symbol,MODE_ASK);
    point=MarketInfo(symbol,MODE_POINT);
    double spread=MarketInfo(symbol,MODE_SPREAD);
    if (point==0.00001){point=0.0001;spread=spread*0.1;}

    if (point==0.001){point=0.01;spread=spread*0.1;}
    double Price=OrderOpenPrice();
    double sp=0;
    if (OrderType()==OP_BUY) {sp=(bid-Price)/point;if(sp>0)Spribl+=sp;}
    if (OrderType()==OP_SELL){sp=(Price-ask)/point;if(sp>0)Spribl+=sp;}
    }

return(Spribl);
}
double MMinus()
{
int total=OrdersTotal();string symbol="";
double point=0.0001;double Spribl=0;
for (int i=0;i<total; i++)
   {if(OrderSelect(i, SELECT_BY_POS)==true)
    symbol=OrderSymbol();
    double bid=MarketInfo(symbol,MODE_BID);
    double ask=MarketInfo(symbol,MODE_ASK);
    point=MarketInfo(symbol,MODE_POINT);
    double spread=MarketInfo(symbol,MODE_SPREAD);
    if (point==0.00001){point=0.0001;spread=spread*0.1;}
    if (point==0.001){point=0.01;spread=spread*0.1;}
    double Price=OrderOpenPrice();
    double sp=0;
    if (OrderType()==OP_BUY){sp=(bid-Price)/point;if(sp<-v8)Spribl=MathMin(sp,Spribl);}
    if (OrderType()==OP_SELL){sp=(Price-ask)/point;if(sp<-v8)Spribl=MathMin(sp,Spribl);}
    }
 if (Spribl==0)Spribl=1;
return(Spribl);
}
double Minus()
{
int total=OrdersTotal();string symbol="";
double point=0.0001;double Spribl=0;
int m=0;
for (int i=0;i<total; i++)
   {if(OrderSelect(i, SELECT_BY_POS)==true)
    symbol=OrderSymbol();
    for(int j=1;j<=10;j++)if (symbol==SIM[j])m=j;
    double bid=MarketInfo(symbol,MODE_BID);
    double ask=MarketInfo(symbol,MODE_ASK);
    point=MarketInfo(symbol,MODE_POINT);
```

```
    double spread=MarketInfo(symbol,MODE_SPREAD);
    if (point==0.00001){point=0.0001;spread=spread*0.1;}
    if (point==0.001){point=0.01;spread=spread*0.1;}
    double Price=OrderOpenPrice();
    double sp=0;
    if (OrderType()==OP_BUY){sp=(bid-Price)/point;if(sp<0)Spribl+=MathAbs(sp);}
    if (OrderType()==OP_SELL){sp=(Price-ask)/point;if(sp<0)Spribl+=MathAbs(sp);}
    }

return(Spribl);
}
int Kplus()
{
int total=OrdersTotal();string symbol="";
double point=0.0001;int Spribl=0;
int m=0;
for (int i=0;i<total; i++)
   {if(OrderSelect(i, SELECT_BY_POS)==true)
    symbol=OrderSymbol();
    for(int j=1;j<=10;j++)if (symbol==SIM[j])m=j;
    double bid=MarketInfo(symbol,MODE_BID);
    double ask=MarketInfo(symbol,MODE_ASK);
    point=MarketInfo(symbol,MODE_POINT);
    double spread=MarketInfo(symbol,MODE_SPREAD);
    if (point==0.00001){point=0.0001;spread=spread*0.1;}
    if (point==0.001){point=0.01;spread=spread*0.1;}
    double Openprice=OrderOpenPrice();
    if (OrderType()==OP_BUY&& (bid-Openprice)/point>0)Spribl++;
    if (OrderType()==OP_SELL&&(Openprice-ask)/point>0)Spribl++;
    }

return(Spribl);}

int Kminus()
{
int total=OrdersTotal();string symbol="";
double point=0.0001;int Spribl=0;

int m=0;
for (int i=0;i<total; i++)
   {if(OrderSelect(i, SELECT_BY_POS)==true)
    symbol=OrderSymbol();
    for(int j=1;j<=10;j++)if (symbol==SIM[j])m=j;
    double bid=MarketInfo(symbol,MODE_BID);
    double ask=MarketInfo(symbol,MODE_ASK);
    point=MarketInfo(symbol,MODE_POINT);
    double spread=MarketInfo(symbol,MODE_SPREAD);
    if (point==0.00001){point=0.0001;spread=spread*0.1;}
    if (point==0.001){point=0.01;spread=spread*0.1;}
    double Openprice=OrderOpenPrice();
    if (OrderType()==OP_BUY&& (ask-Openprice)/point<0)Spribl++;
    if (OrderType()==OP_SELL&&(Openprice-bid)/point<0)Spribl++;
    }
```

```
return(Spribl);
}

double Histor()
{
int total=OrdersHistoryTotal();string symbol="";
double point=0.0001;
double Spribl=0;
for (int i=0;i<=total; i++)
   {if(OrderSelect(i, SELECT_BY_POS,MODE_HISTORY)==true)
    symbol=OrderSymbol();
    double bid=MarketInfo(symbol,MODE_BID);
    double ask=MarketInfo(symbol,MODE_ASK);
    double Price=OrderOpenPrice();
    double close=OrderClosePrice();
    if (OrderType()==OP_BUY)Spribl+=(close-Price)/point;
    if (OrderType()==OP_SELL){Spribl+=(Price-close)/point;}
    }
return(Spribl);
}
double LIMIT()
{
int total=OrdersTotal();
double Spribl=0;
for (int i=0;i<total; i++)
   {if(OrderSelect(i, SELECT_BY_POS)==true)
    if (OrderType()==OP_BUYLIMIT)Spribl++;
    if (OrderType()==OP_SELLLIMIT)Spribl++;
    if (OrderType()==OP_BUY)Spribl++;
    if (OrderType()==OP_SELL)Spribl++;
    }

return(Spribl);
}
int Logsim(string SIMic,int z)
{
int total=OrdersTotal();
int shet=0;
for (int i=0;i<total; i++)
   {if(OrderSelect(i, SELECT_BY_POS)==true&&OrderSymbol()==SIMic)
    {if (OrderType()==OP_BUY&&z==1)shet++;
    if (OrderType()==OP_SELL&&z==1)shet++;
    if (OrderType()==OP_BUYLIMIT&&z==2)shet++;
    if (OrderType()==OP_SELLLIMIT&&z==2)shet++;}
    }

return(shet);
}
int Logall(int z)
{
int total=OrdersTotal();
int shet=0;
   for (int i=0;i<total; i++)
   {if(OrderSelect(i, SELECT_BY_POS)==true)
```

```
    {if (OrderType()==OP_BUY&&z==1)shet++;
    if (OrderType()==OP_SELL&&z==1)shet++;
    if (OrderType()==OP_BUYLIMIT&&z==2)shet++;
    if (OrderType()==OP_SELLLIMIT&&z==2)shet++;}
    }

return(shet);
}
int DELALL()
{
int total=OrdersTotal();string symbol="";
for (int i=0;i<total; i++)
   {if(OrderSelect(i, SELECT_BY_POS)==true)
    symbol=OrderSymbol();
    double bid=MarketInfo(symbol,MODE_BID);
    double ask=MarketInfo(symbol,MODE_ASK);
    if (OrderType()==OP_BUY)if(!OrderClose(OrderTicket(),OrderLots(),bid,1,0))
            Print("OrderClose error ",GetLastError());
    if (OrderType()==OP_SELL)if(!OrderClose(OrderTicket(),OrderLots(),ask,1,0))
            Print("OrderClose error ",GetLastError());
    if (OrderType()==OP_BUYLIMIT)
    if(!OrderDelete(OrderTicket()))Print("OrderClose1 error ",GetLastError());
    if (OrderType()==OP_SELLLIMIT)
    if(!OrderDelete(OrderTicket()))Print("OrderClose1 error ",GetLastError());
    }
Print(" DELALL");
return(0);
}
int DELSIM(string SIMic)
{
int total=OrdersTotal();
for (int i=0;i<total; i++)
   {if(OrderSelect(i, SELECT_BY_POS)==true&&OrderSymbol()==SIMic)
      {double bid=MarketInfo(SIMic,MODE_BID);double ask=MarketInfo(SIMic,MODE_ASK);
          if (OrderType()==OP_BUY)if(!OrderClose(OrderTicket(),OrderLots(),bid,1,0))
            Print("OrderClose1 error ",GetLastError());
          if (OrderType()==OP_SELL)if(!OrderClose(OrderTicket(),OrderLots(),ask,1,0))
            Print("OrderClose1 error ",GetLastError());}
    }
    Print(SIMic," DELSIM");
return(0);
}
int DELTICK(string SIMic)
{
int total=OrdersTotal();
int tick=-1;int shet=0;
for (int i=0;i<total; i++)
   {
   if(OrderSelect(i, SELECT_BY_POS)==true&&OrderSymbol()==SIMic)
   if (OrderType()==OP_BUY||OrderType()==OP_SELL)
   {shet++;if (tick<0)tick=OrderTicket();else tick=MathMin(OrderTicket(),tick);}
   }
if (shet<2)return(0);
for (int i=0;i<total; i++)
```

```
   {if(OrderSelect(i, SELECT_BY_POS)==true&&OrderSymbol()==SIMic&&OrderTicket()==tick)
      {double bid=MarketInfo(SIMic,MODE_BID);double ask=MarketInfo(SIMic,MODE_ASK);
    if (OrderType()==OP_BUY)if(!OrderClose(OrderTicket(),OrderLots(),bid,1,0))
            Print("OrderClose1 error ",GetLastError());
    if (OrderType()==OP_SELL)if(!OrderClose(OrderTicket(),OrderLots(),ask,1,0))
            Print("OrderClose1 error ",GetLastError());}
    }
    Print(SIMic," DELTICK");
return(0);
}
int DELLIMIT(string SIMic)
{
int total=OrdersTotal();string symbol="";
for (int i=0;i<total; i++)
   {if(OrderSelect(i, SELECT_BY_POS)==true)
    symbol=OrderSymbol();
    double bid=MarketInfo(symbol,MODE_BID);
    double ask=MarketInfo(symbol,MODE_ASK);
    if (OrderType()==OP_BUYLIMIT&&OrderSymbol()==SIMic)
    if(!OrderDelete(OrderTicket()))Print("OrderClose1 error ",GetLastError());
      else return(1);
    if (OrderType()==OP_SELLLIMIT&&OrderSymbol()==SIMic)
    if(!OrderDelete(OrderTicket()))Print("OrderClose1 error ",GetLastError());
      else return(1);
    }

return(0);
}
int DELSELLLIMIT(string SIMic)
{
int total=OrdersTotal();string symbol="";
for (int i=0;i<total; i++)
   {if(OrderSelect(i, SELECT_BY_POS)==true)
    symbol=OrderSymbol();
    double bid=MarketInfo(symbol,MODE_BID);
    double ask=MarketInfo(symbol,MODE_ASK);
    if (OrderType()==OP_SELLLIMIT&&OrderSymbol()==SIMic)
    if(!OrderDelete(OrderTicket()))Print("OrderClose1 error ",GetLastError());
    }

return(0);
}
int DELBUYLIMIT(string SIMic)
{
int total=OrdersTotal();string symbol="";
for (int i=0;i<total; i++)
   {if(OrderSelect(i, SELECT_BY_POS)==true)
    symbol=OrderSymbol();
    double bid=MarketInfo(symbol,MODE_BID);
    double ask=MarketInfo(symbol,MODE_ASK);
    if (OrderType()==OP_BUYLIMIT&&OrderSymbol()==SIMic)
    if(!OrderDelete(OrderTicket()))Print("OrderClose1 error ",GetLastError());
    }
```

```
return(0);
}
int Modi(string SIMic, int j)
{
int total=OrdersTotal();
double point=MarketInfo(SIMic,MODE_POINT);int okr=4;
if (point==0.00001)point=0.0001;
if (point==0.001)point=0.01;
if (point==0.01)okr=2;
double ask=MarketInfo(SIMic,MODE_ASK);
double bid=MarketInfo(SIMic,MODE_BID);
double spread=1+(ask-bid)/point;
spread=MathMax(3,spread);
double buystoploss=NormalizeDouble(bid-MathMax(v15,MathAbs(Aib[j]))*point,okr);
double sellstoploss=NormalizeDouble(ask+MathMax(v15,MathAbs(Aib[j]))*point,okr);

bool res=0;
for (int i=0;i<total;i++)
   {
  if(OrderSelect(i, SELECT_BY_POS)==true && OrderSymbol()==SIMic)
    {
   if(OrderType()==OP_BUY)
     if
(buystoploss>OrderStopLoss())res=OrderModify(OrderTicket(),OrderOpenPrice(),buystoploss,Order
TakeProfit(),0);
    if(OrderType()==OP_SELL)
      if
(sellstoploss<OrderStopLoss()||OrderStopLoss()==0)res=OrderModify(OrderTicket(),OrderOpenPric
e(),sellstoploss,OrderTakeProfit(),0);

    }
    }
return(2);
}

int ModiTake(string SIMic,int j)
{

int total=OrdersTotal();
double point=MarketInfo(SIMic,MODE_POINT);int okr=4;
if (point==0.00001)point=0.0001;
if (point==0.001)point=0.01;
if (point==0.01)okr=2;
double ask=MarketInfo(SIMic,MODE_ASK);
double bid=MarketInfo(SIMic,MODE_BID);
double buytake=NormalizeDouble(bid+Aib[j]*v28/v15*point,okr);
double selltake=NormalizeDouble(ask-Aib[j]*v28/v15*point,okr);

bool res=0;
for (int i=0;i<total;i++)
   {
  if(OrderSelect(i, SELECT_BY_POS)==true && OrderSymbol()==SIMic)
    {
   if(OrderType()==OP_BUY)
```

```
     if
(buytake<OrderTakeProfit())res=OrderModify(OrderTicket(),OrderOpenPrice(),OrderStopLoss(),buy
take,0);
    if(OrderType()==OP_SELL)
     if
(selltake>OrderTakeProfit())res=OrderModify(OrderTicket(),OrderOpenPrice(),OrderStopLoss(),sellt
ake,0);
    }
    }
return(2);
}
double MODILIMIT(string SIMic,int j)
{
int total=OrdersTotal();
double point=MarketInfo(SIMic,MODE_POINT);int okr=4;
if (point==0.00001)point=0.0001;
if (point==0.001)point=0.01;
if (point==0.01)okr=2;
double ask=MarketInfo(SIMic,MODE_ASK);
double bid=MarketInfo(SIMic,MODE_BID);
double Openprice=0;
bool res=0;
for (int i=0;i<total;i++)
   {
  if(OrderSelect(i, SELECT_BY_POS)==true && OrderSymbol()==SIMic)
    {
    if(OrderType()==OP_BUYLIMIT)
         {if(!OrderModify(OrderTicket(),OrderOpenPrice(),
         NormalizeDouble(ask-StopLost*point,okr),
         OrderTakeProfit(),0,0))
         Print(SIMic," Error modifications1 orders. OP_BUYLIMIT. Error code=",GetLastError());
         else Print(SIMic," modifications1 orders OP_BUYLIMIT");}

    if(OrderType()==OP_SELLLIMIT)
         {if(!OrderModify(OrderTicket(),OrderOpenPrice(),
         NormalizeDouble(bid+StopLost*point,okr),
         OrderTakeProfit(),0,0))
         Print(SIMic, "Error order modifications. OP_SELLLIMIT. Error code=",GetLastError());
         else Print(SIMic, "modifications1 orders OP_SELLLIMIT");}


    }
    }
return(0);
}
double MODILIMIT2(string SIMic,int j)
{
int total=OrdersTotal();
double point=MarketInfo(SIMic,MODE_POINT);int okr=4;
if (point==0.00001)point=0.0001;
if (point==0.001)point=0.01;
if (point==0.01)okr=2;
double ask=MarketInfo(SIMic,MODE_ASK);
double bid=MarketInfo(SIMic,MODE_BID);
```

```
double Openprice=0;
bool res=0;
for (int i=0;i<total;i++)
  {
  if(OrderSelect(i, SELECT_BY_POS)==true && OrderSymbol()==SIMic)
   {

   if(OrderType()==OP_BUYLIMIT&&NormalizeDouble(ask-
forlimit[j]*point,okr)<OrderOpenPrice())
        {
        if(!OrderModify(OrderTicket(),NormalizeDouble(ask-forlimit[j]*point,okr),
        NormalizeDouble(ask-StopLost*point,okr),
        OrderTakeProfit(),0,0))
        Print(SIMic,NormalizeDouble(ask-StopLost*point,okr)," Error order modifications.
OP_BUYLIMIT. Error code=",GetLastError());
        else {Print(SIMic," order modifications OP_BUYLIMIT");return(1);}}


if(OrderType()==OP_SELLLIMIT&&NormalizeDouble(bid+forlimit[j]*point,okr)>OrderOpenPrice
())
        {if(!OrderModify(OrderTicket(),NormalizeDouble(bid+forlimit[j]*point,okr),
        NormalizeDouble(bid+StopLost*point,okr),
        OrderTakeProfit(),0,0))
        Print(SIMic, NormalizeDouble(ask-StopLost*point,okr)," Error order modifications.
OP_SELLLIMIT. Error code=",GetLastError());
        else {Print(SIMic, "order modifications OP_SELLLIMIT");return(1);}}

   }
   }
return(0);
}
double MODILIMIT3(string SIMic,int j)
{
int total=OrdersTotal();
double point=MarketInfo(SIMic,MODE_POINT);int okr=4;
if (point==0.00001)point=0.0001;
if (point==0.001)point=0.01;
if (point==0.01)okr=2;
double ask=MarketInfo(SIMic,MODE_ASK);
double bid=MarketInfo(SIMic,MODE_BID);
double Openprice=0;
bool res=0;
for (int i=0;i<total;i++)
  {
  if(OrderSelect(i, SELECT_BY_POS)==true && OrderSymbol()==SIMic)
   {

   if(OrderType()==OP_BUYLIMIT&&NormalizeDouble(ask-
forlimit[j]*point,okr)>OrderOpenPrice())
        {
        if(!OrderModify(OrderTicket(),NormalizeDouble(ask-forlimit[j]*point,okr),
        NormalizeDouble(ask-StopLost*point,okr),
        OrderTakeProfit(),0,0))
        Print(SIMic,NormalizeDouble(ask-StopLost*point,okr)," Error modifications orders.
```

```
OP_BUYLIMIT. Error code=",GetLastError());
        else Print(SIMic," modifications orders OP_BUYLIMIT");}


if(OrderType()==OP_SELLLIMIT&&NormalizeDouble(bid+forlimit[j]*point,okr)<OrderOpenPrice
())
        {if(!OrderModify(OrderTicket(),NormalizeDouble(bid+forlimit[j]*point,okr),
        NormalizeDouble(bid+StopLost*point,okr),
        OrderTakeProfit(),0,0))
        Print(SIMic, NormalizeDouble(ask-StopLost*point,okr)," Error modifications orders.
OP_SELLLIMIT. Error code=",GetLastError());
        else Print(SIMic, "modifications orders OP_SELLLIMIT");}

    }
    }
return(0);
}
double MODI3(string SIMic,int j)
{
int total=OrdersTotal();
double point=MarketInfo(SIMic,MODE_POINT);int okr=4;
if (point==0.00001)point=0.0001;
if (point==0.001)point=0.01;
if (point==0.01)okr=2;
double ask=MarketInfo(SIMic,MODE_ASK);
double bid=MarketInfo(SIMic,MODE_BID);
double spread=(ask-bid)/point;
double takeprofit=0;
bool res=0;
for (int i=0;i<total;i++)
   {
   if(OrderSelect(i, SELECT_BY_POS)==true && OrderSymbol()==SIMic)
    {
    if(OrderType()==OP_BUY)
        {
        takeprofit=NormalizeDouble(OrderOpenPrice()+(Aib[j]+profit[j])*point,okr);
        if ((bid-takeprofit)/point>spread)takeprofit=NormalizeDouble(ask+spread*point,okr);
        if(!OrderModify(OrderTicket(),OrderOpenPrice(),
        OrderStopLoss(),
        takeprofit,0,0))
        Print(SIMic," Error order modifications. OP_BUYLIMIT. Error code=",GetLastError());
        else Print(SIMic," order modifications MODI3");}

    if(OrderType()==OP_SELL)
        {
        takeprofit=NormalizeDouble(OrderOpenPrice()-(Aib[j]+profit[j])*point,okr);
        if ((takeprofit-ask)/point>spread)takeprofit=NormalizeDouble(bid-spread*point,okr);
        if(!OrderModify(OrderTicket(),OrderOpenPrice(),
        OrderStopLoss(),
        takeprofit,0,0))
        Print(SIMic, "Error order modifications. OP_SELLLIMIT. Error code=",GetLastError());
        else Print(SIMic, "order modifications MODI3");}
    }
    }
```

```
return(0);
}

double v()
{
int total=OrdersTotal();
double sr=0;double ss=0;
for (int i=0;i<=total;i++)
if(OrderSelect(i, SELECT_BY_POS)==true)
   {if(OrderType()==OP_SELL)
   {for(int j=1;j<=10;j++)
      if(SIM[j]==OrderSymbol())
      {sr+=MathMax(Aib[j],v15);
      ss++;}
   }
   if(OrderType()==OP_BUY)
   {for(int j=1;j<=10;j++)
      if(SIM[j]==OrderSymbol())
      {sr+=MathMax(Aib[j],v15);
      ss++;}
   }
   }
if (ss==0)return(v15);
return(sr/ss);
}

int End()
{
int total=OrdersHistoryTotal();
string symbol="";string symbol1="";string symbol2="";
double point=0.0001;
double Spribl=0;double Sprib2=0;
for (int i=0;i<=total; i++)
   {if(OrderSelect(i, SELECT_BY_POS,MODE_HISTORY)==true)
    symbol=OrderSymbol();
    double close=OrderClosePrice();
    double TakeProfit=OrderTakeProfit();
    double Profit=OrderProfit();
    if (close==TakeProfit&&Profit>0){symbol1=symbol;Spribl=Profit;}
    if (Profit!=0){symbol2=symbol;Sprib2=Profit;}
    }
 if (Spribl!=0
      &&Sprib2!=0
        &&symbol1==symbol2
          &&symbol1!=""
            &&Spribl==Sprib2
             &&Spribl>0)
 return(1);

return(0);
}
```